ESEIAAT

Projecte de Final de Grau

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

---

Estudi i aplicació de tècniques de control robust a la navegació d'un quadrotor

---

Memòria

Grau: Grau en Enginyeria en Vehicles Aeroespacials
Data d'entrega: 2018 - 01 - 10
Estudiant: Irene Pérez Martínez
Director: Albert Masip Álvarez

Resum

En aquest estudi es parteix d'un treball ja realitzat amb un vehicle aeri no tripulat o *Dron* quadrotor [9] governat per la seva sensòrica de posició, velocitat, acceleració i càmara de vídeo, i programat amb eines com MATLAB i ROS. S'estudia més a fons la dinàmica d'aquesta aeronau plantejant primer el model analític, però resolent-ho després amb dades experimentals. Per aconseguir-ho, s'apliquen tècniques d'identificació robusta desenvolupades per Levy, Harris, Ljung i Bhattacharyya entre d'altres. A més, es plantegen algoritmes per assolir-ne el control robust i poder aplicar-ho al seguiment d'un circuit pintat al terra, mitjançant la càmara zenital del *Dron*.
L'objectiu final per a l'estudiant és el d'aprendre un procediment d'identificació en el domini freqüencial, que és aplicable a qualsevol sistema, robot o màquina. A més, ha servit per iniciar-se en la disciplina del control automàtic i els sistemes reals.

Agraïments

A l'Albert, per no parar d'animar, ensenyar, guiar, de treure's alternatives de la màniga, i de fer que aquest treball hagi estat una experiència única.
Al David, per confiar en mi i per totes les oportunitats que m'ha donat aquest any.
Als meus pares, Eduard i Anna, pel suport financer i la cultura de l'esforç que m'han inculcat.
Al Genís, pel suport moral, alimentari i per haver-se llegit el treball N vegades.
A la Sandra i l'Alberto, per fer divertida i familiar la vida al TR11.
Als meus amics físics, per no deixar de pressionar-me, més o menys afectuosament, perquè acabi alguna carrera.

MOLTES GRÀCIES A TOTS!.

Índex

Índex de figures

Índex de taules

Part I

Introducció

Capítol 1

Introducció

Avui en dia una branca important de la ciència i tecnologia es troben enfocades a l'autonomia de les màquines per a que puguin desenvolupar un servei a les persones. En el cas que ocupa aquest estudi, els *Drons*, s'intenta aportar un granet de sorra a l'automatització d'aquestes màquines programant-hi algoritmes que, a través d'una càmera de vídeo, els facin seguir circuits pintats al terra.
És difícil no haver sentit a parlar de l'ús militar d'aquestes aeronaus, guiades per coordenades GPS, que ja fa temps que s'utilitzen en molts països, i la tecnologia dels quals s'està aplicant a camps com l'agricultura, medi ambient, centrals elèctriques, etc. El nou repte es presenta quan es vol fer que aquestes aeronaus siguin "intel·ligents".
Les teories que s'utilitzaran en aquest estudi provenen de la branca del control automàtic, des de les més bàsiques sobre interpretació de dades d'un experiment fins a tècniques de control robust desenvolupades en els últims 30 anys. Les eines que s'utilitzaran provenen del software mundialment reconegut i emprat per enginyers com és MATLAB[1], i del programari lliure ROS[2], una plataforma per a crear aplicacions de robots totalment gratuïta i creada per al lliure desenvolupament de tecnologia arreu del món.

1.1 Objectius

L'obejctiu d'aquest estudi consisteix en dur a terme la identificació robusta del model dinàmic que descriu el comportament cinemàtic d'un *Dron* quadrotor, analitzat en funció de les consignes d'angle. Partint d'aquest model, serà necessari sintetitzar els paràmetres dels controladors robusts utilitzant tècniques polinomials. La utilització d'aquests controladors permetrà controlar la trajectòria del *Dron* mitjançant l'adquisició i processat d'imatges de la càmara que duu integrada.
El hardware de l'estudi consisteix en un *Parrot AR.Drone 2.0*, i la corresponent comunicació via WiFi amb l'ordinador utilitzant MATLAB i ROS. Aquest *Dron* ja disposa d'un controlador intern desconegut, que serà combinat amb els nous models i controladors. Aquest estudi realitzarà diferents tests enviant consignes d'angle i registrant resultats de velocitat lineal, ja que l'objectiu principal serà aprendre i aplicar el procediment d'identificació robusta, de la mateixa manera que es pot aplicar a altres vehicles aeris.

[1] MATrix LABoratory
[2] Robot Operating System

1.2 Abast

Al principi del projecte es va determinar el següent llistat de tasques a desenvolupar:

1. **Posada a punt de la plataforma:** Cercar estudis previs al voltant d'aquest *Dron*, establir les connexions via ROS i MATLAB per communicar-s'hi i realitzar maniobres d'enlairament i aterratge amb l'aeronau. A més, s'ha d'estudiar les possibilitats de consignes per enviar que poden ser útils per a desenvolupar la identificació robusta. Practicar amb el generador de MATLAB de senyals multisinus.

2. **Identificació (un model):** Realitzar i classificar diferents tests enviant senyals multisinus rics en freqüències a les consignes dels tres angles de moviment: *pitch, roll* i *yaw*. Analitzar la resposta en velocitat lineal del sistema en el domini freqüencial, i ajustar les dades recollides a un model basat en una funció de transferència (quocient entre un numerador i denominador polinomials). Validar-lo en el domini temporal.

3. **Disseny del controlador de velocitat lineal:** Calcular la funció de transferència d'un controlador utilitzant el mètode de síntesi directa. Establir un llaç tancat entre les consignes d'angle i les lectures de velocitat lineal, tant en simulació amb Simulink com provant-ho al *Dron*.

4. **Adquisició i processat de les imatges de la càmara zenital:** Subscriure's al canal d'imatge de les dues càmeres, adquirir les imatges en format *raw*, filtrar-les aplicant bases colorimètriques i un filtre blanc i negre de MATLAB amb la finalitat de detectar el color vermell i la forma de les línies pintades al terra.

5. **Disseny del controlador de trajectòria:** Tancar el llaç més extern amb l'objectiu d'utilitzar les imatges processades com a "outputs" i consignes de la posició, i complir la missió de fer el seguiment de les línies del terra.

6. **Identificació robusta:** Trobar les debilitats del model sintetitzat a la tasca 2 d'acord amb les limitacions del *hardware*, i ajustar nous models diferents per guanyar robustesa a la identificació del comportament cinemàtic del *Dron*.

7. **Re-disseny dels controladors:** Com s'ha procedit a les tasques 3 i 5, es recalculen els controladors sobre la velocitat lineal i sobre la trajectòria per als nous models trobats a la tasca 6.

1.3 Motivació

El sector de les aeronaus no tripulades és una tecnologia relativament nova, amb un creixement espectacular en els últims temps, i amb múltiples aplicacions. Si bé fins al moment les més directes han estat orientades a pilotar-los des d'una estació remota, s'estan destinant molts recursos i esforços a fer que aquests vehicles siguin autònoms.
La recerca en aquest camp combina experiments sobre una aeronau, fer funcionar un hardware mecànic i de comunicacions que pot resultar complicat, realitzar anàlisis sobre dades experimentals, programació d'algoritmes d'identificació, programació d'algoritmes de moviment i planificació i, per acabar, veure com vola el *Dron*! No és d'extranyar doncs, que aquest projecte resultés prometedor per a l'estudiant des d'un bon principi, fos un repte en moltes disciplines que encara no havia treballat, i obrís la porta a seguir un possible camí dins la branca del control automàtic.

1.4 Estat de l'art

Els quadrotors tenen molts avantatges davant els helicòpters degut a la seva simplicitat mecànica ja que no necessiten d'una articulació especial per variar l'angle de pitch cíclic o col·lectiu de les pales. El seu disseny, muntatge i manteniment són senzills, i si les seves dimensions són petites, resulten relativament segurs per als humans. No poden transportar-ne però poden realitzar nombrosos serveis. Avui en dia però, l'atenció dels vehicles aeris no tripulats es centra en el control autònom amb propòsits militars, científics o recreacionals.

Per al propòsit d'aquest estudi s'ha triat el vehicle *Parrot AR.Drone 2.0* ja que promet una plataforma senzilla i intuitiva de connectar, utilitzar i comunicar-se amb l'ordinador. Aquest vehicle ja ha estat utilitzat en estudis previs de la UPC, com el treball *Control de trajectòria en un Dron UAV (2016)*, que ha servit de guia per a aquest estudi des del principi. No obstant, aquest vehicle no es troba ja a la venda, al haver estat substituït per altres models com Bebop, Mambo o Disco que incorporen millors mecanismes, càmeres i control per ulleres de FPV[3] entre d'altres.
A més, tot el coneixement aplicat a l'estudi prové de classes de la UPC a les que l'estudiant ha assistit, com per exemple *Control Automàtic*, o s'ha après durant el període de desenvolupament del projecte, majoritàriament provinent del director del projecte.

[3]FPV: First Person View

Capítol 2

Posada a punt de la plataforma

Per a la realització d'aquest projecte s'ha utilitzat tres eines bàsiques: ROS i MATLAB instal·lats en un PC ASUS amb sistema operatiu Ubuntu Mate 16.04 i el Dron Parrot AR.Drone 2.0.

2.1 Robot Operating System

ROS es una plataforma flexible per desenvolupar software de robots, de programari lliure. És un conjunt d'eines, llibreries i convencions amb el propòsit de simplificar la tasca de crear aplicacions per a robots per a una àmplia varietat de plataformes.
ROS treballa creant una xarxa de nodes entre tots els sistemes que participen en una aplicació, dels quals el coordinador o *master* és l'ordinador de dins del robot i la resta són nodes. La comunicació entre ells es pot establir de dues maneres, enviant un missatge en un instant determinat d'un node a un altre, o creant un canal (Tòpics) on un dels nodes publiqui contínuament informació i els altres nodes s'hi subscriguin i puguin rebre'l. Es pot veure en l'esquema següent:

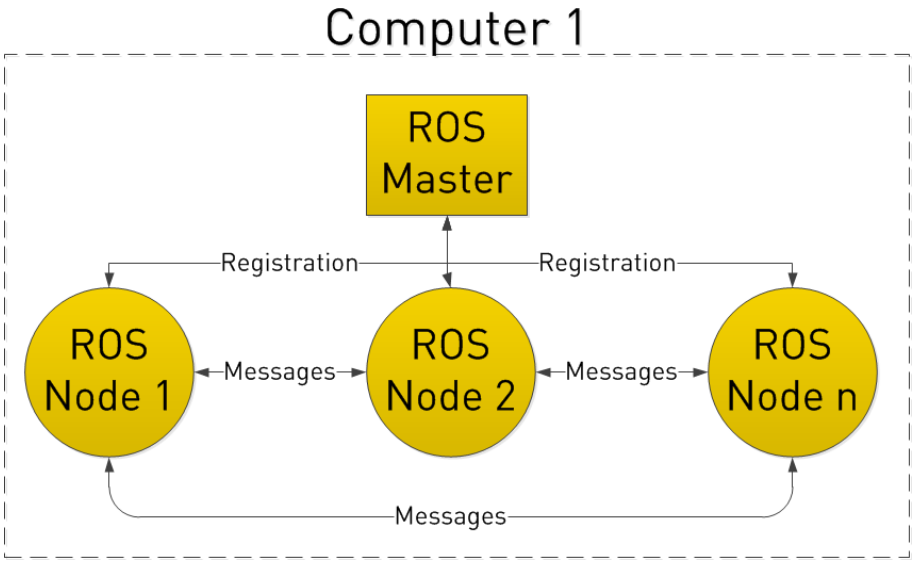


Figura 2.1: *Esquema de funcionament de ROS*

Un node és, a la pràctica, un fitxer executable dins del paquet de ROS. Els nodes utilitzen les llibreries de ROS per comunicar-se amb altres nodes. Aquests nodes poden publicar o subscriure's a un Tòpic, o proporcionar o utilitzar un Servei.

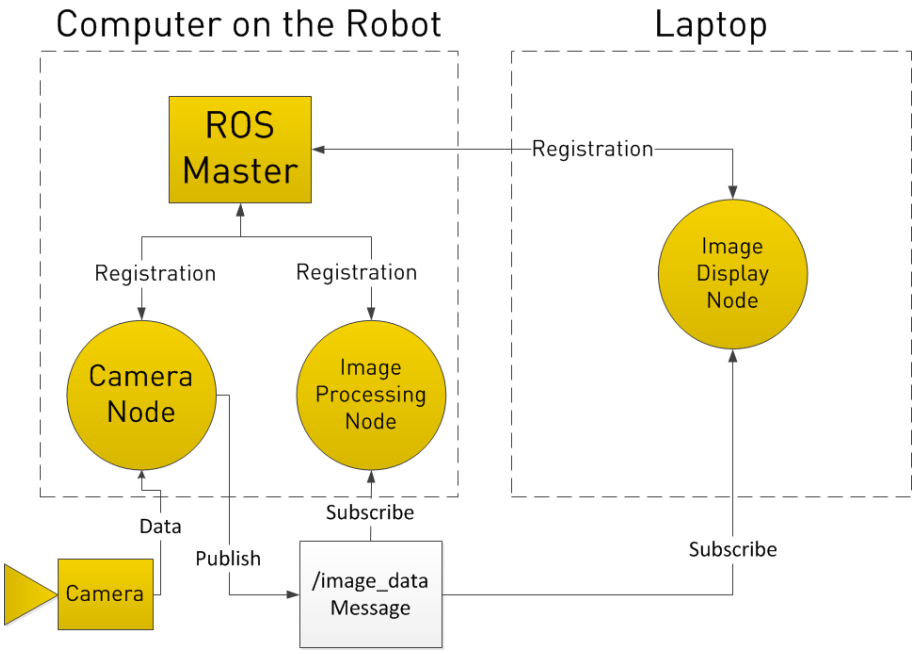


Figura 2.2: *Esquema de funcionament de ROS*

En aquest PC ASUS, de processador Intel Core i7 de 8 GB de RAM, hi ha instal·lat el sistema operatiu *Ubuntu MATE 16.04*, i la versió r2017b de MATLAB, i s'ha triat l'última versió LTS [1] de ROS, *Kinetic Kame*, llançada el 23 de Març de 2016.

Figura 2.3: *Distribució de ROS*

Un cop instal·lada aquesta versió, i com que ROS és un sistema operatiu que respon a línies de comandes *Bash*, es poden enviar ordres des del terminal de l'ordinador. Així, per inicialitzar-lo i executar la controladora del *Dron* cal declarar

```
rosrun arDron_autonomy arDron_driver
```

Les llibreries de ROS permeten que els nodes es comuniquin en dos llenguatges de programació diferents: Python (`rospy`) i C++ (`roscpp`), i per interactuar-hi s'utilitza MATLAB.

[1] LTS: Long Term Support

2.2 MATLAB

MATLAB (acrònim de LABoratori de MATrius) és un entorn de computació i un llenguatge de programació en sí mateix, que va ser creat en el seu origen per enginyers de control que van fundar l'empresa *Mathworks* al 1984. Permet crear algoritmes de computació numèrica, mostrar gràfics i imatges, i comunicar-se amb altres llenguatges de programació.
El suport de MATLAB per ROS és encapsulat en una *Toolbox* (en català una caixa d'eines) anomenada *Robotic Systems Toolbox*, una llibreria de funcions que permet intercanviar dades amb robots dirigits per ROS. Desde línia de comandes de MATLAB, si es crida

```
rosinit
```

es crea un ROS *master* dins d'aquest entorn, i un node global connectat amb ell. Des d'aquest node *master*, es coordinen tota la resta de parts de la xarxa. S'identifica amb el *Master URI*[2] que especifica l'adreça IP de la màquina on s'està executant el node global. Es pot desconnectar fent

```
rosshutdown
```

2.3 Dron Parrot AR.Drone 2.0

Un *Dron* és un vehicle aeri no tripulat (en anglès UAV [3]) que pot ser pilotat per control remot o per instruccions prèviament programades i implementades en la seva controladora. La seva forma i configuració pot ser molt diversa, però en el cas del present estudi, s'ha adquirit un Parrot AR.Drone 2.0 per diferents motius:

- Configuració de quadcòpter
- Fàcil connexió via WiFi
- Kit de desenvolupament de software de Parrot (en anglès SDK [4]) accessible i senzill d'utilitzar
- Preu raonable dins del pressupost del treball
- Càmara zenital per implementar el seguiment de línia

[2]URI: Uniform Resource Identifier
[3]UAV: Unmanned Aerial Vehicle
[4]SDK: Software Development Kit

Figura 2.4: *Parrot AR.Drone 2.0*

2.3.1 Especificacions tècniques del *Dron*

El vehicle *Parrot AR.Drone 2.0* va ser llançat amb fins recreacionals, a un preu assequible i amb bones prestacions tècniques. Encara es pot adquirir des de la web de Parrot o a través de portals de segona mà. La següent informació tècnica s'ha extret del mateix lloc web de Parrot:

1. Grabació de vídeo HD
   - Càmera HD 720p 30 fps
   - Objectiu gran angular amb diagonal de 92◦
   - Perfil de codificació bàsica H264
   - Format JPEG de les fotografies
   - Connexió WiFi
2. Assistència electrònica
   - Processador d'1 GHz 32 bits ARM Cortex A8 amb DSP vídeo 800 MHz TMS320DMC64x
   - Sistema operatiu Linux 2.6.32
   - Memòria RAM DDR2 d'1 GB a 200 MHz
   - USB 2.0 d'alta velocitat per a les extensions
   - WiFi bgn
   - Giròscop de 3 eixos, amb precisió de 2000° per segon
   - Acceleròmetre de 3 eixos, amb precisió de ±50 mg
   - Magnetòmetre de 3 eixos, amb precisió de 6°
   - Sensor de pressió amb precisió de ± 10 Pa
   - Sensors d'ultrasons per mesurar l'altitud
   - Càmera vertical QVGA 60 fps per mesurar la velocitat durant el vol
3. Bateries
   - Bateries LiPo de 1000 mAh
4. Motorització
   - 4 motors sense escombretes del tipus "inrunner": 14,5 W i 28500 rpm controlats per un micro-controlador
   - Coixinets de boles en miniatura
   - Engranatges de Nylatron
   - Coixinets de bronze auto-lubricants
5. Pes
   - Amb carcassa d'interior pesa 380 g
   - Amb carcassa d'exterior pesa 420 g

2.3.2 Integració del *Dron* amb ROS

Sembla clar ja quina informació es vol enviar o extreure del *Dron* a través del PC. Com ja s'ha comentat, la comunicació entre aquests dos nodes s'estableix a través de canals publicadors i subscriptors de ROS, per tant es detallarà les comandes exactes de MATLAB que s'executen quan es vol governar el *Dron* des del PC:

- Publicador per a resetejar el *Dron*:

```
resetPUB = rospublisher('/arDron/reset')
magnetPUB = rospublisher('/arDron/mag')
```

- Missatge buit per a diferents tòpics:

```
emptyMSG = rosmessage('std_msgs/Empty')
```

- Client per resetejar els sensors:

```
flattrimCLI = rossvcclient('/arDron/flattrim')
```

- Client per seleccionar o canviar de càmera:

```
setCamFrontCLI = rossvcclient('/arDron/setcamchannel')
toggleCamCLI = rossvcclient('/arDron/togglecam')
```

- Publicadors de control del *Dron* per enlairar-se i aterrar:

```
takeOffPUB = rospublisher('/arDron/takeoff')
landPUB = rospublisher('/arDron/land')
```

- Publicador de comandes de moviment, i missatge amb les comandes concretes:

```
cmd_velPUB = rospublisher('cmd_vel')
geometryMSG = rosmessage('geometry_msgs/Twist');
geometryMSG.Linear.X= 0;
geometryMSG.Linear.Y= 0;
geometryMSG.Linear.Z= 0;
geometryMSG.Angular.X= 0;
geometryMSG.Angular.Y= 0;
geometryMSG.Angular.Z= 0;
```

- Subscriptor de les dades de navegació a través dels sensors:

```
navdataSUB = rossubscriber('/arDron/navdata')
```

Aquest és tot el "vocabulari" necessari per comunicar-se. Cada vegada que s'executa una rutina des de MATLAB cal inicialitzar tots aquests canals. Després, per a que el *Dron* dugui a terme les actuacions desitjades cal, en primer lloc, enlairar-lo:

```
send(takeOffPUB,emptyMSG)
```

enviar-li el missatge amb les consignes de moviment i publicar-lo al canal corresponent:

```
send(cmd_velPUB, geometryMSG)
```

rebre les imatges zenitals del canal de la càmara:

```
imatge = receive(imageSUB)
```

i rebre els valors mesurats pels sensors de les variables de moviment:

```
navdata = receive(navdataSUB)
```

Finalment, quan es desitgi o s'acabi la rutina, cal fer aterrar el *Dron*:

```
send(landPUB, emptyMSG)
```

Capítol 3

Marc teòric: Mecànica del vol del *Dron*

Un quadcòpter obté la sustentació per volar i maniobrar de 4 rotors disposats en creu i equidistants del centre de l'aeronau. Aquests rotors poden girar en sentit horari o antihorari i a diferents velocitats per modificar la trajectòria del *Dron*, així, si per exemple un parell oposat de rotors giren en sentit horari i l'altre parell en sentit anti-horari es diu que realitza un vol a punt fix (en anglès *Hovering*). Per definir la resta de moviments s'ha d'establir el sistema d'eixos de referència en el que es mou, i considerar que és un instrument amb 6 graus de llibertat:

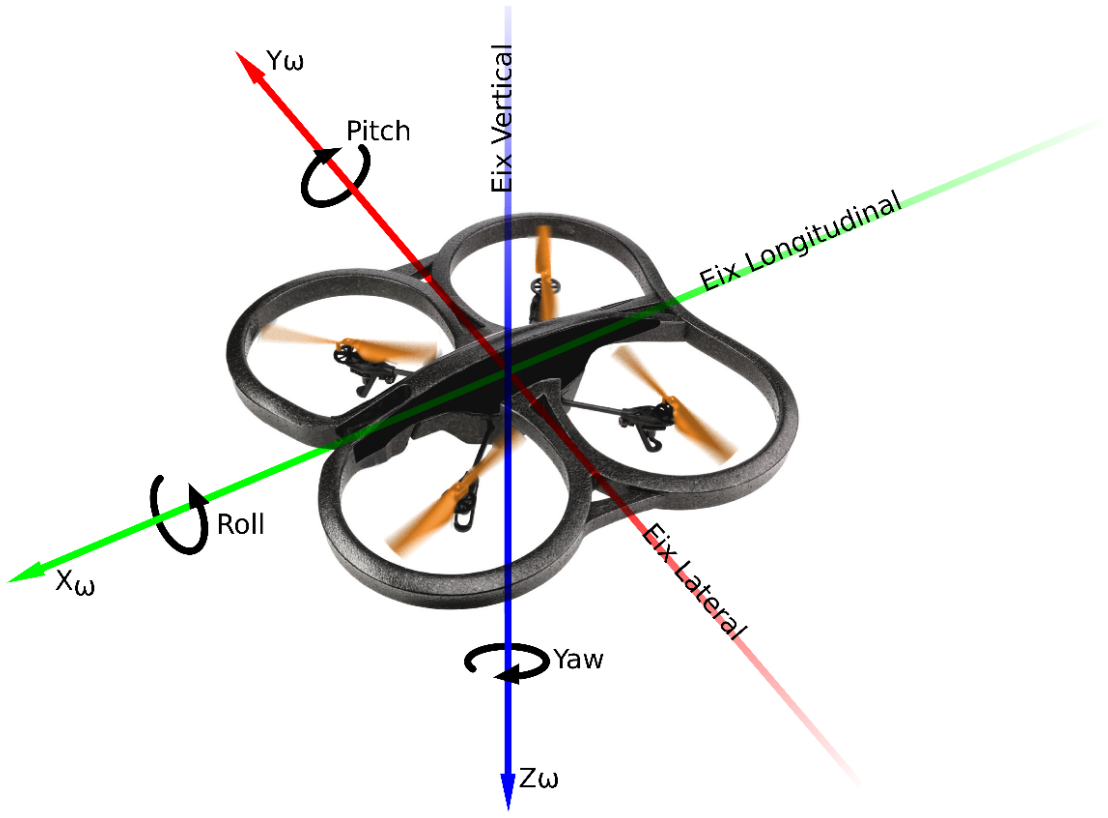


Figura 3.1: *Sistema d'eixos centrat en el c.d.g. de l'aeronau*

El sistema d'eixos horitzó local té l'origen en el centre de gravetat de l'aeronau i la orientació igual que la dels eixos terra, l'eix x_h dirigit al nord, y_h a l'est i z_h al centre de la Terra. El

sistema d'eixos "cos"també té origen en el centre de gravetat de l'aeronau però l'eix x_b apunta al "morro" i z_b pertany al pla de simetria de l'aeronau. Així, per definir-ne el moviment, es consideraran tres angles entre aquests dos sistemes de referència com a graus de llibertat, els anomentats **Angles d'Euler**:

- Angle de capcineig ($-\frac{\pi}{2} < \theta < \frac{\pi}{2}$): Rotació sobre l'eix lateral (en anglès *Pitch*)
- Angle de balanceig ($-\pi < \phi < \pi$): Rotació sobre l'eix longitudinal (en anglès *Roll*)
- Angle de guinyada ($0 < \psi < 2\pi$): Rotació sobre l'eix vertical (en anglès *Yaw*)

Des del punt de vista pràctic, la controladora del *Dron* entén els graus de llibertat com a *Pitch-Roll-Yaw*, i se li enviaran les consignes d'aquests angles reescalades en tant per cent. Addicionalment, la controladora contempla els canvis d'altitud com a *Gaz*.

3.1 Equacions generals del moviment

3.1.1 Sistemes de referència

Per estudiar la cinemàtica i dinàmica de l'aeronau s'utilitzen els dos sistemes de referència esmentats, que permeten simplificar les equacions generals del moviment. La transformació d'un a l'altre es realitza mitjançant les matrius de rotació:

1. Guinyada sobre l'eix z:
$$\begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
2. Capcineig sobre l'eix y:
$$\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$
3. Balanceig sobre l'eix x:
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$$

En total, la matriu de canvi d'eixos queda com:

$$L_{bh} = \begin{bmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix}$$

3.1.2 Relacions dinàmiques

Partim del teorema de la quantitat de moviment

$$\vec{F} = \frac{d(m\vec{V})}{dt} \tag{3.1}$$

i del teorema de moment cinètic

$$\vec{G} = \frac{d(I\vec{\omega})}{dt} \tag{3.2}$$

on $\vec{F}$ és el vector de forces externes que actuen sobre l'aeronau (F_x, F_y, F_z), $\vec{V}$ és el vector de velocitats absolutes del centre de gravetat de l'aeronau (u, v, w), $\vec{G}$ és el vector de moments externs al voltant del centre de gravetat (L, M, N), I és el tensor d'inèrcia i $\vec{\omega}$ és el vector de velocitats angulars absolutes (p, q, r).
Desenvolupant l'equació (3.1) s'ha de tenir en compte que F s'expressa en eixos "cos" i V en eixos terra, un sistema d'eixos inercial. Per això, al fer la derivada s'utilitza l'operador derivada en base mòbil:

$$\vec{F} = m\left(\frac{\partial \vec{V}}{\partial t} + \vec{\omega} \times \vec{V}\right) \tag{3.3}$$

Així s'obté un sistema de tres equacions per a les forces externes tal que

$$\begin{Bmatrix} F_x \\ F_y \\ F_z \end{Bmatrix} = m\left[\begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{Bmatrix} + \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ p & q & r \\ u & v & w \end{vmatrix}\right] \tag{3.4}$$

Per utilitzar l'equació (3.2) cal saber que el tensor d'inèrcia en una aeronau essencialment simètrica en els tres eixos no té les components creuades J_{yz}, J_{xz}, i J_{xy} i queda

$$\vec{h} = I\vec{\omega} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} = \begin{Bmatrix} I_x p \\ I_y q \\ I_z r \end{Bmatrix} \tag{3.5}$$

Així, s'obté un sistema de tres equacions per als moments externs tal que

$$\begin{Bmatrix} L \\ M \\ N \end{Bmatrix} = \frac{\partial \vec{h}}{\partial t} + \vec{\omega} \times \vec{h} = \begin{Bmatrix} I_x\dot{p} + (I_z - I_y)qr \\ I_y\dot{q} - (I_z - I_x)pr + \\ I_z\dot{r} - (I_x - I_y)pq \end{Bmatrix} \tag{3.6}$$

3.1.3 Relacions cinemàtiques angulars

Les velocitats angulars poden posar-se en funció dels angles d'Euler:

$$\vec{\omega}_{bh} = \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} = \dot{\psi} L_{bh} \vec{k_h} + \dot{\theta} L_{b1} \vec{j_1} + \dot{\phi} \vec{i_b} \tag{3.7}$$

Així el sistema queda

$$\begin{aligned} p &= \dot{\phi} - \dot{\psi}\sin\theta \\ q &= \dot{\theta}\cos\phi + \dot{\psi}\sin\phi\cos\theta \\ r &= -\dot{\theta}\sin\phi + \dot{\psi}\cos\phi\cos\theta \end{aligned} \tag{3.8}$$

3.1.4 Relacions cinemàtiques lineals

Les velocitats lineals ja es troben en eixos "cos" , però és interessant expressar-les en eixos terra (igualment orientats que els eixos horitzó local amb la hipòtesi de terra plana [1]) per poder-les integrar i trobar les trajectòries reals descrites per l'aeronau.

$$\vec{V}\,)_h = \begin{Bmatrix} \dot{x_h} \\ \dot{y_h} \\ \dot{z_h} \end{Bmatrix} = L_{hb}\vec{V}\,)_b = L_{bh}^T \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{3.9}$$

$$\vec{V}\,)_h = \begin{Bmatrix} u\cos\theta\cos\psi + v(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + w(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ u\cos\theta\sin\psi + v(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) + w(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \\ -u\sin\theta + v\sin\phi\cos\theta + w\cos\phi\cos\theta \end{Bmatrix} \tag{3.10}$$

3.1.5 Accions externes

Les forces externes es poden separar en les de caràcter aerodinàmic, propulsives i gravitatòries: $\vec{F} = \vec{F}_A + \vec{F}_T + \vec{F}_G$.
Les expressades en eixos terra són les gravitatòries (el pes), per tant s'hauran de transformar amb la matriu L_{bh}:

$$\vec{F}_G\,)_h = \begin{Bmatrix} 0 \\ 0 \\ mg \end{Bmatrix} \quad \longrightarrow \quad \vec{F}_G\,)_b = L_{bh}\vec{F}_G\,)_h = \begin{Bmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\cos\phi \end{Bmatrix} \tag{3.11}$$

Les forces propulsives provenen de l'empenta proporcionada pels quatre motors, així, la força resultant en la direcció z_b serà $F_T = F_1 + F_2 + F_3 + F_4$.

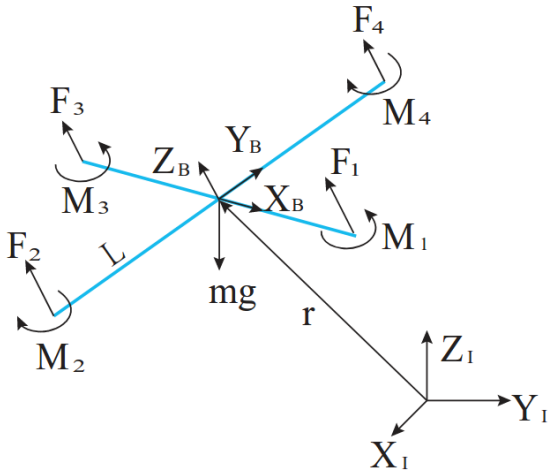


Figura 3.2: *Forces en un quadrotor UAV*

[1] Hipòtesi de terra plana: a alçades petites comparades amb el radi de la Terra es considera que el sistema d'eixos terra és inercial

Considerant que els 4 braços són iguals i de longitud l, la rotació dels motors també produeix un moment de gir en els tres sentits, que corresponen als tres moments angulars descrits en (3.6) (L, M, N):

$$\tau_\phi = l(F_3 - F_1) \qquad \tau_\theta = l(F_4 - F_2) \qquad \tau_\psi = l(F_2 + F_4 - F_1 - F_3) \tag{3.12}$$

Les forces aerodinàmiques s'expressen en eixos vent com:

$$\vec{F}_A\,)_v = \begin{Bmatrix} -D \\ -Q \\ -L \end{Bmatrix} \tag{3.13}$$

3.1.6 Sistema de forces resultant

Reagrupant totes les expressions anteriors ((3.10), (3.4), (3.8), (3.6)), es pot veure com s'expressa la dinàmica de l'aeronau en funció dels angles i velocitats angulars i lineals descrits.

$$\boxed{\begin{array}{c} \dot{x_h} = u\cos\theta\cos\psi + v(\sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi) + w(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi) \\ \dot{y_h} = u\cos\theta\sin\psi + v(\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi) + w(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi) \\ \dot{z_h} = -u\sin\theta + v\sin\phi\cos\theta + w\cos\phi\cos\theta \end{array}}$$

$$\boxed{\begin{array}{c} F_x = m(\dot{u} - rv + qw) \\ F_y = m(\dot{v} + ru - pw) \\ F_z = m(\dot{w} - qu + pv) \end{array}}$$

$$\boxed{\begin{array}{c} p = \dot{\phi} - \dot{\psi}\sin\theta \\ q = \dot{\theta}\cos\phi + \dot{\psi}\sin\phi\cos\theta \\ r = -\dot{\theta}\sin\phi + \dot{\psi}\cos\phi\cos\theta \end{array}}$$

$$\boxed{\begin{array}{c} l(F_3 - F_1) = I_x\dot{p} + (I_z - I_y)qr \\ l(F_4 - F_2) = I_y\dot{q} - (I_z - I_x)pr \\ l(F_1 + F_3 - F_2 - F_4) = I_z\dot{r} - (I_x - I_y)pq \end{array}}$$

Equacions en eixos vent

Per simplificar les equacions finals, s'adaptarà tot a aquest sistema de referència que té l'origen en el centre de gravetat de l'aeronau, l'eix x_v en la direcció del vent aigües amunt i l'eix z_v dins del pla de simetria de l'avió. En aquest cas, els angles d'orientació canvien de nomenclatura a

$$\begin{Bmatrix} \psi \\ \theta \\ \phi \end{Bmatrix} \longrightarrow \begin{Bmatrix} \chi \\ \gamma \\ \mu \end{Bmatrix} \tag{3.14}$$

i les forces propulsives adquireixen angle d'atac i lateral

$$F_T \longrightarrow \begin{Bmatrix} F_T\cos\epsilon\cos\nu \\ F_T\cos\epsilon\sin\nu \\ -F_T\sin\epsilon \end{Bmatrix} \tag{3.15}$$

Finalment, el sistema d'equacions finals que regeix qualsevol possible moviment de l'aeronau és

$$\begin{array}{c} F_T \cos\epsilon \cos\nu - D - mg\sin\gamma - m\dot{u} = 0 \\ F_T \cos\epsilon \sin\nu - Q + mg\cos\gamma \sin\mu + mu(\dot{\gamma}\sin\mu - \dot{\chi}\cos\gamma\cos\mu) = 0 \\ -F_T \sin\epsilon - L + mg\cos\gamma\cos\mu + mu(\dot{\gamma}\cos\mu + \dot{\chi}\cos\gamma\sin\mu) = 0 \end{array}$$

3.1.7 Altres consideracions

En aquest model no es consideren efectes en les pales com els de *flapping*[2] o *lead lag* [3], degut a la seva complexitat i ordre elevat.
En altres estudis [17] es linearitza el model trobat en l'espai d'estats mitjançant aproximacions de Taylor al primer ordre. L'objectiu d'aquest estudi, com es discutirà més endavant, no és el de realitzar aproximacions, sinó d'identificar completa i experimentalment la dinàmica del sistema.

3.2 Actuacions

Donats els corresponents angles d'Euler, i segons les revolucions a les que giri cada rotor, l'aeronau pot realitzar 5 moviments diferents:

- *Hovering*: els 4 motors giren a les mateixes r.p.m., un parell oposat en sentit horari i l'altre parell en sentit anti-horari. La força de sustentació que generen els 4 motors és la mateixa, i totes sumades compensen el pes de l'aeronau per a què es mantingui sempre a la mateixa alçada.

- Balanceig: per aconseguir un angle de *Roll* positiu, el parell de motors de la dreta de l'eix x_b disminueixen les seves r.p.m.s. L'aeronau s'inclina i guanya velocitat lineal en y.

- Guinyada: per aconseguir un angle de *Yaw* positiu, partint de la configuració de *Hovering*, s'incrementen les r.p.m.s del parell de motors que giraven en sentit horari. La sustentació no canvia, per tant, les r.p.m.s totals no s'han d'incrementar.

- Canvi d'altitud: per aconseguir que el *Dron* pugi o baixi, n'hi ha prou amb accelerar o desaccelerar tots els motors alhora perquè guanyi o perdi sustentació.

- Capcineig: per aconseguir un angle de *Pitch* negatiu, els dos motors davanters giren a menys r.p.m.s que els darrers. Així, l'aeronau s'inclina endavant un angle θ i la propulsió s'inverteix en sustentar i en accelerar en la direcció x. Com que el pes no canvia, s'ha d'incrementar les r.p.m.s totals per sustentar igual que abans.

3.3 Simplificació del model

D'acord amb la teoria que s'ha desenvolupat en el capítol anterior, el sistema sembla fortament acoblat, i pot resultar difícil fer-se una idea de quines són les dependències entre variables. Malgrat això, el *Dron* ja porta de fàbrica un controlador intern que fa que aquest acoblament no es manifesti sensiblement. De cara a les equacions, es poden assumir algunes hipòtesis per

[2] *flapping*: batiment de les pales
[3] *lead lag*: arrossegament de les pales

Figura 3.3: *Sentit de gir dels motors en capcineig*

simplificar-les i veure'n les dependències.
Essencialment, el sistema és un sòlid rígid de massa m amb 4 braços suspesa a l'aire per l'acció de 4 rotors que provoquen una empenta en la direcció vertical. La primera simplificació serà suposar que els angles són petits (θ i ϕ), i que es fixarà l'angle de guinyada ($\psi = 0$). Així, la matriu de canvi d'eixos L_{bh} només dependrà de θ i ϕ i, al derivar-la, se'n pot negligir els termes no lineals. Per tant l'equació (3.10) queda:

$$\begin{Bmatrix} \ddot{x_h} \\ \ddot{y_h} \\ \ddot{z_h} \end{Bmatrix} = L_{bh}^{T,simplificada} \begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{Bmatrix} \tag{3.16}$$

i l'equació (3.8) queda:

$$\begin{aligned} p &= \dot{\phi} \\ q &= \dot{\theta} \\ r &= \dot{\phi} \end{aligned} \tag{3.17}$$

Negligint els termes de Coriolis de l'equació (3.6) queda:

$$\begin{aligned} l(F_3 - F_1) &= I_x \dot{p} \\ l(F_4 - F_2) &= I_y \dot{q} \\ l(F_1 + F_3 - F_2 - F_4) &= I_z \dot{r} \end{aligned} \tag{3.18}$$

i els termes petits de l'equació (3.4) queda:

$$\begin{Bmatrix} F_x \\ F_y \\ F_z \end{Bmatrix} = m \begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{Bmatrix} \tag{3.19}$$

on, en una simplificació més, només es tindran en compte el pes i la força d'empenta total dels motors.
Reagrupant-les totes quatre, i expressant-les en eixos terra, es pot veure la dependència directa de les trajectòries que descriu el *Dron* (acceleracions en els tres eixos) amb els tres angles de moviment, que multipliquen als moments d'inèrcia i que es troben en la matriu de canvi d'eixos.

$$\boxed{\begin{array}{c} l(F_3 - F_1) = I_x\ddot{\phi} \\ l(F_4 - F_2) = I_y\ddot{\theta} \\ l(F_1 + F_3 - F_2 - F_4) = I_z\ddot{\psi} \end{array}}$$

$$\boxed{\begin{Bmatrix} \ddot{x_h} \\ \ddot{y_h} \\ \ddot{z_h} \end{Bmatrix} = \frac{1}{m}\left[\begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} + L_{bh}^{T,simplificada} \cdot \begin{pmatrix} 0 \\ 0 \\ -(F_1 + F_2 + F_3 + F_4) \end{pmatrix}\right]}$$

Part II

Modelització i Control

Capítol 4

Identificació d'un model

4.1 Què és la identificació?

'La **Identificació** d'un sistema és l'art i ciència de construir models matemàtics sobre sistemes dinàmics a partir de dades empíriques d'entrada i sortida ' [16].

La **Modelització**, en canvi, fa ús de les equacions dinàmiques (diferencials) que neixen de les lleis físiques que expliquen el comportament del procés.
En el capítol 4 s'ha arribat a l'expressió d'un model per a entendre la dinàmica d'aquest sistema. En altres estudis s'ha linearitzat models similars fent aproximacions de primer ordre, els quals són acurats si els angles d'actuació són petits, i pels quals és més fàcil dissenyar controladors. No obstant, 'tractant el AR.Dron com a una caixa negra i caracteritzant-ne les entrades i sortides, queda inherentment modelada la dinàmica tant complexa de l'aparell. Per exemple, queda contemplat el temps de resposta dels motors, el *flapping* de les pales, l'efecte giroscòpic sobre els rotors, forces d'inèrcia, resistència aerodinàmica, etc' [17].

4.2 Quin és l'objectiu d'aquesta identificació?

Es pretén que el *Dron* reconegui línies i cruïlles de color vermell pintades al terra mitjançant la càmera zenital. Per a aquest propòsit donar-li consignes d'angle de *Pitch* perquè voli endavant seguint la línia, consignes de *Roll* perquè intenti centrar-se a sí mateix sobre la línia, i consignes de *Yaw* per centrar-se i realitzar girs en arribar a una cruïlla.
En aquest aspecte, el llaç de control intern del sistema del *Dron* sobre les consignes d'angle és opac i l'única sortida que es proporciona és el canal *navdata* cridat des de MATLAB. En aquest canal es reben els valors 'reals' (calculats amb els sensors de l'aeronau) dels tres angles, alçada, velocitats lineals en x i y i velocitat angular en z.
Per al propòsit d'aquest estudi s'envien, per tant, consignes d'angle de *Pitch, Roll* i *Yaw* que es transformen a velocitats lineals en x, y i angular en z.

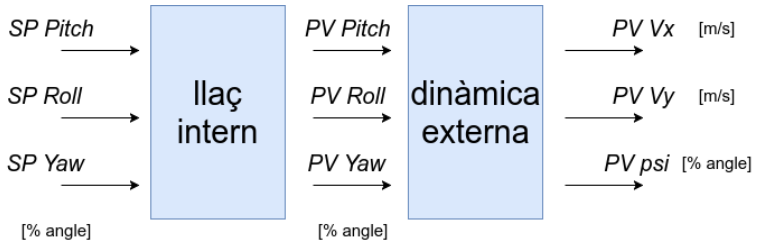


Figura 4.1: *Esquema d'entrades i sortides dels assajos*

El Parrot, al ser un producte comercial ja acabat, ja disposa d'un llaç intern per controlar els angles. Com que l'eina *navdata* proporciona velocitats de sortida s'identificarà la resposta d'aquestes velocitats davant l'entrada d'angles de *Pitch, Roll* i *Yaw*. És el que s'identifica en la figura seguent com a G_1:

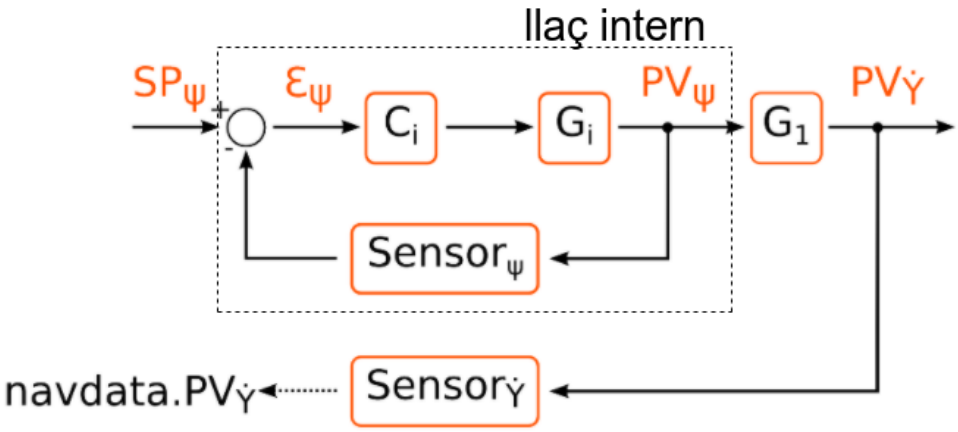


Figura 4.2: *Imatge extreta de [9]*

4.3 Per què resposta en freqüència?

La identificació d'un sistema es pot realitzar en el domini temporal o en el domini freqüencial, tenint en compte si el model es vol estimar en temps continu o discret.
En aquest estudi, s'ha apostat pel domini freqüencial ja que, com s'exposa a l'article de Lennard Ljung [15], aquesta posa de manifest comportaments dinàmics (apreciables en les dades empíriques) del procés que ajuden a l'hora de proposar l'ordre del model. Aquesta identificació ja no resulta un problema per a l'estimació de models en temps continu, ja que per aconseguir dades limitades en un rang concret s'utilitzaran finestres. 'La transformada discreta de Fourier (DFT) de l'entrada serà igual a la transformada de Fourier de l'entrada temporal contínua associada' [15].
A més, amb aquest mètode és possible estimar la incertesa associada al soroll de les dades, expressant els paràmetres del model com a intervals, i traduint-ho a una identificació anomenada **Identificació Robusta**.

4.4 Rutina utilitzada

Per executar el *Dron* i realitzar el seguiment de línia s'utilitzen rutines de MATLAB heredades del treball previ [9] i depurades (per exemple, eliminant variables globals):

- `ARD_ROS.m`: inicialitza els canals de ROS, enlaira i fa aterrar el *Dron*, i declara el temporitzador `ROStimer.m`
- `ROStimer.m`: aquest temporitzador és una funció que es repeteix cada 0.1 segons. Inclou la crida del canal d'imatge i del canal de dades de navegació
- `TractaImatge.m`: realitza el tractament d'imatge i retorna els punts de les línies detectades
- `Controlador.m`: rep la posició de les línies i calcula consignes d'angle de *Pitch, Roll, Yaw* i *Gaz* per enviar-les al *Dron*

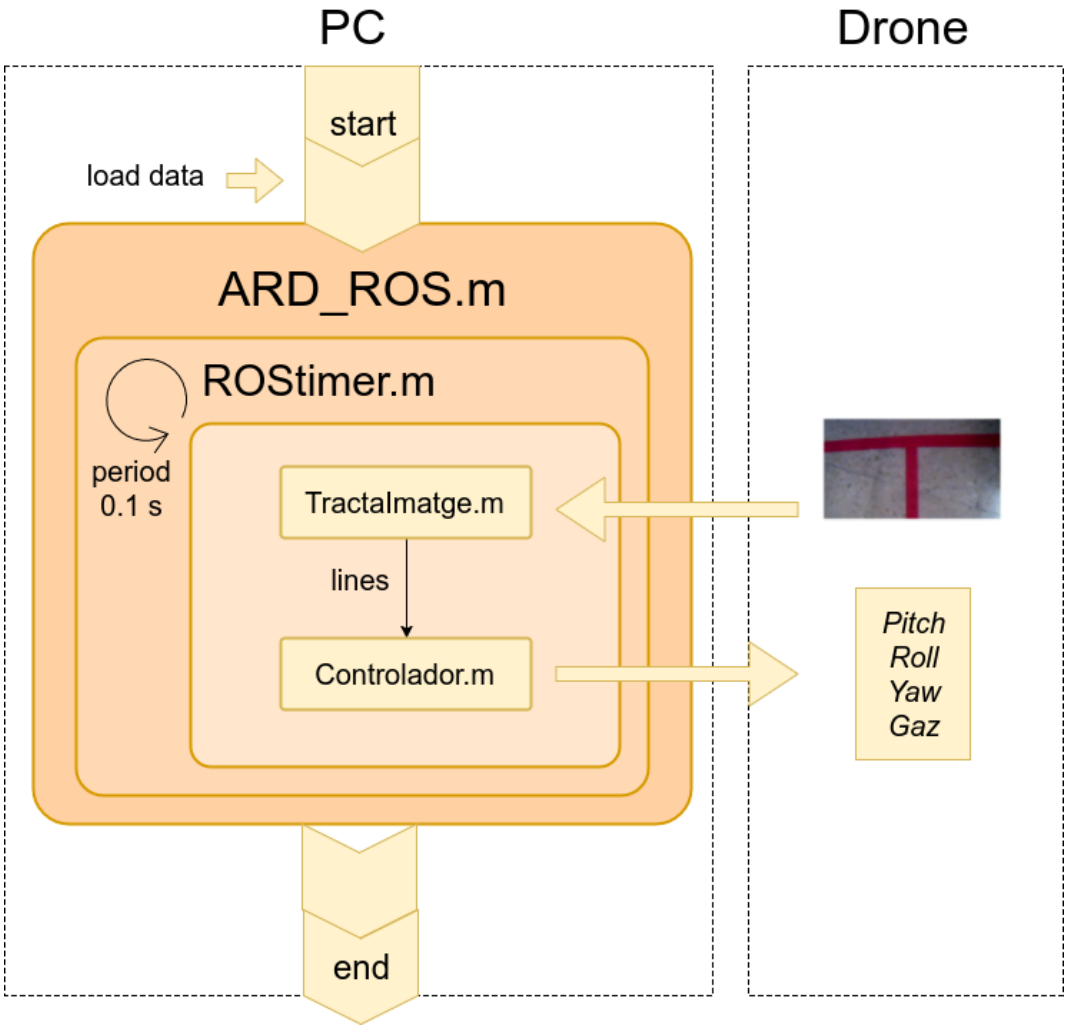


Figura 4.3: *Esquema de funcionament del programa d'assajos*

4.4.1 Justificació del temps de mostreig i d'enviament de dades

L'eina `tic - toc` de MATLAB permet saber el temps que es triga en realitzar una rutina, amb precisió de dècimes de mil·lisegon. S'ha utilitzat per a calcular els temps dels dos processos importants que es duen a terme en el codi:

	Tractament d'imatge	**Dades de navegació**	**Total de la rutina**
Mitjana	0.0684 s	0.0160 s	0.0830 s
Desviació estàndard	0.0242 s	0.0099 s	0.0255 s

Taula 4.1: *Temps en un assaig de 150 segons en posició de Hovering*

Cada X temps se li ha d'enviar les noves consignes de moviment al *Dron*, i ha de ser capaç de processar la imatge que rep i calcular els valors d'angle, com es detalla en la taula anterior. Per tant, com que el temps de mostreig ha de ser superior al temps total de la rutina, s'agafarà 0.1 segons.

4.5 Identificació de la dinàmica del *Dron*

4.5.1 Dades d'entrada dels assajos

Per al quadrotor els graus de llibertat amb què es pretén controlar les velocitats en x i y són: *Pitch, Roll, Yaw, Gaz*. Per dissenyar-ne controladors primer s'ha d'identificar els paràmetres dels models d'aquests angles.
El present estudi se centrarà en el *Pitch* i el *Roll*, se n'identificarà els models realitzant diferents assajos en un espai tancat. Aquests assajos es classifiquen segons el grau de llibertat, el període de mostreig i la magnitud. Així, per exemple

- Assaig $r10\_005.mat$:
  - r: *Roll*
  - 10: magnitud de 1.0 (en %)
  - 005: temps de mostreig de 0.05 segons
- Assaig $p01\_01.mat$:
  - p: *Pitch*
  - 01: magnitud de 0.1 (en %)
  - 01: temps de mostreig de 0.1 segons

Senyal multisinus

En les tècniques d'estimació de resposta freqüencial s'intenta enviar senyals d'entrada el més rics harmònicament possible per a què reflexin el comportament no lineal d'un sistema. Quan el sistema es pot descriure per un model simple d'ordre baix es pot enviar un graó, rampa o

paràbola. No obstant, com més freqüències exciti l'entrada més dades es tindran per la identificació posterior, per tant s'estudia enviar un senyal que sigui la suma de funcions sinusoidals amb diferents freqüències:

$$x(t) = \sum_{i=1}^{N} a_i \cos(2\pi f_i t) \tag{4.1}$$

El problema d'aquests senyals transitoris és que donen amplituds molt altes. La solució la proposen algoritmes que busquen minimitzar el quocient entre l'amplitud més alta i el *RMS*[1] del senyal generat, anomenat factor de cresta.

$$cr = \frac{X_{max}}{X_{rms}} \tag{4.2}$$

Així, el senyal multisinus triat es crea amb l'eina `msinclip2.m` introduint-li un vector amb 15 freqüències desitjades, el temps de mostreig (període de repetició del programa `ROStimer.m`), el nombre de punts, i el mínim i màxim valor de consigna que es vol prendre:

```
f = linspace(.1,2,15)

[time,data,matriu]=Generador(f,ts,np,v,V)

[time,data,matriu]=Generador(f,0.1,50,-0.1,0.1)
```

El senyal obtingut amb aquest mètode `Generador` s'ha enviat com a consigna dels tres angles de *Pitch, Roll* i *Yaw*.

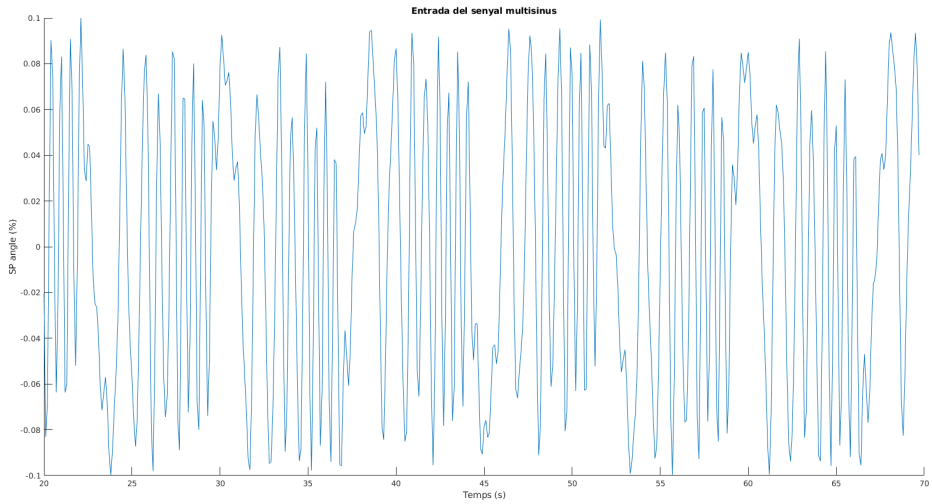


Figura 4.4: *Dades d'entrada en domini temporal*

[1]RMS: root mean square (en català és la mitjana quadràtica)

Finestra de Blackmann Harris

L'estimació d'un senyal correspon a obtenir els valors dels paràmetres que descriuen aquest senyal. Com que el senyal amb què s'excita el *Dron* és periòdic, es descomposa en funcions elementals com sinus i cosinus mitjançant la transformada de Fourier (FFT). [2]
No obstant, sorgeix un problema important en la identificació mitjançant aquest procediment: una línia freqüencial "forta" pot afectar la detecció d'una altra línia freqüencial més dèbil que sigui propera a la primera. El soroll que es genera al voltant del pic més alt pot fer que es passi per alt l'efecte del pic més baix. Per aquesta raó, quan es calcula la transformada, s'ha de retallar les dades (altrament dit enfinestrar) de tal manera que es puguin reconèixer suficientment bé les freqüències excitades.
Harris explica en l'article [10] la utilitat d'enfinestrar quan es fan anàlisis de senyals harmònics. En concret presenta diferents finestres que es poden aplicar per tractar de "netejar" les dades de la transformada. Per exemple, aplicant una finestra rectangular s'obté el resultat següent:

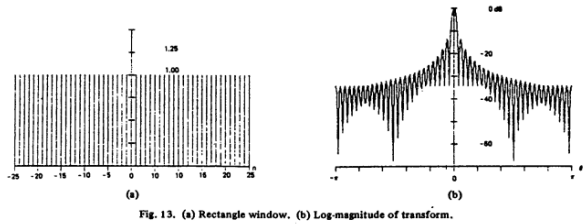


Figura 4.5: *Finestra rectangular sobre un senyal periòdic*

En canvi, si s'aplica una finestra de Blackmann sobre el mateix senyal s'obté el següent:

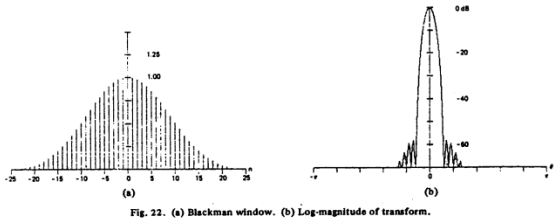


Figura 4.6: *Finestra de Blackmann sobre un senyal periòdic*

L'escollida ha estat la finestra de Blackmann-Harris, pel seu bon comportament als límits de l'interval. Es pot calcular com:

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi}{N}n\right) + a_2 \cos\left(\frac{2\pi}{N}2n\right) - a_3 \cos\left(\frac{2\pi}{N}3n\right) \tag{4.3}$$

on s'eliminen quasi per complet els lòbuls dels extrems:

[2] FFT: Fast Fourier Transform (en català Transformada de Fourier Ràpida)

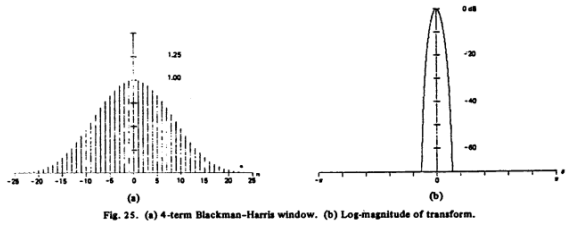
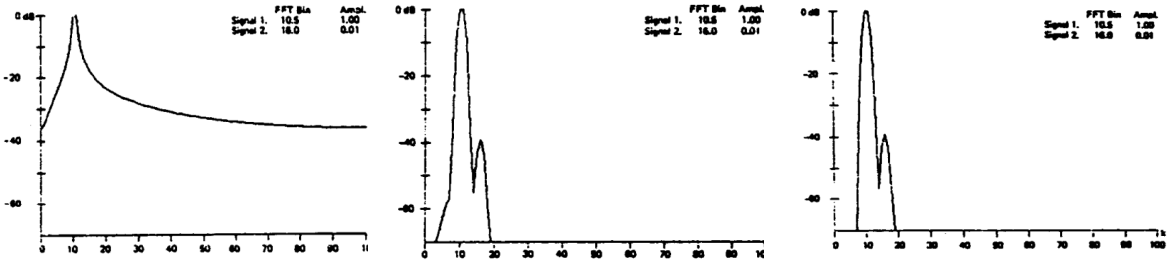


Figura 4.7: *Finestra de Blackmann-Harris sobre un senyal periòdic*

És fàcil veure'n l'efecte sobre un senyal compost per una freqüència "forta" a 10Hz amb amplitud 1, i una freqüència "feble" a 16 Hz amb amplitud 0.01. Es modifica el senyal perquè l'amplitud més gran es doni a 10.5 Hz per exemple. Aleshores es pot veure com responen les 3 finestres anteriors:



(a) *Finestra rectangular* (b) *Finestra de Blackmann* (c) *Finestra Blackmann-Harris*

Figura 4.8: *Finestres aplicades sobre un senyal de freqüències 10 Hz i 16 Hz*

Per tant, a l'aplicar la DFT [3] a les dades de la figura 4.4 enfinestrada s'obté el següent:

```
DiscreteFourierTransform(Entrada,f,.1,1,1,'blackmanharris')
```

[3]DFT: Discrete Fourier Transform (en català Transformada de Fourier Discreta)

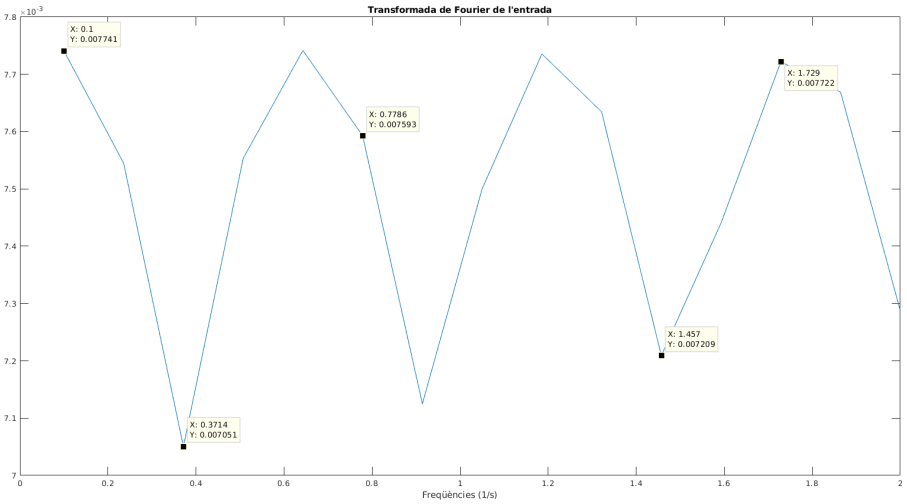


Figura 4.9: *Pics excitats de l'entrada*

Es pot veure en la figura 4.9 que tot i el vector `f` de freqüències enviat, les freqüències excitades han estat les dels 15 primers pics que, amb les transformacions adients es poden identificar com:

`[0.1 0.24 0.37 0.51 0.64 0.78 0.91 1.05 1.19 1.32 1.46 1.59 1.73 1.86 2.00]`.

A partir d'ara s'utilitzaran les dades d'aquests pics ressonants per realitzar la identificació.

4.5.2 Assaig de *Pitch*

El nom de l'assaig per les dades d'entrada serà $p01\_01.mat$, i per les de sortida $p01\_01rX.mat$, on X representa el número d'intents que s'han enregistrat.
S'envia el senyal multisinus de la figura 4.4, però es pren com a sortida la velocitat V_x.

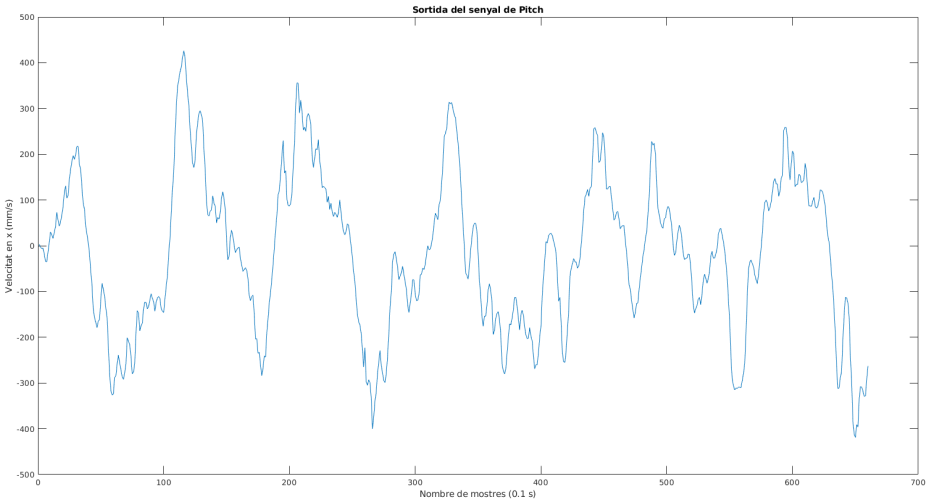


Figura 4.10: *Dades de sortida de l'assaig* $p01\_01r4$ *en domini temporal*

Per identificar el sistema es passa a domini freqüencial, amb posteriors comprovacions en temporal. Les dades de l'ajust correspondran al quocient entre la sortida i l'entrada dels pics seleccionats, és a dir,

$$D(s) = \frac{Fs(s)}{Fe(s)} \tag{4.4}$$

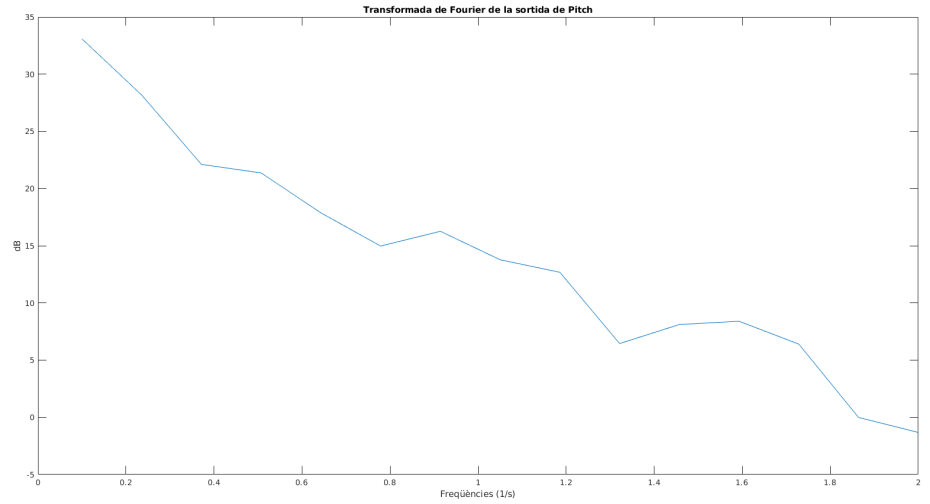


Figura 4.11: *Pics excitats de la sortida l'assaig* $p01\_01r4$

Anàlisi dels resultats

Primerament, s'estima una funció de transferència com a quocient de dos polinomis que poden ser de diferents ordres:

$$FT(s) = \frac{B(s)}{A(s)} \tag{4.5}$$

Mitjançant les dades obtingudes en l'assaig, s'intenta que l'error entre elles i la funció de transferència ajustada sigui mínim o zero:

$$Error(s) = \frac{B(s)}{A(s)} - D(s) \tag{4.6}$$

Per aconseguir-ho, a continuació es detalla el procediment que s'ha seguit, descrit per Levy en el seu article [14]. Ell explica que per ajustar la resposta freqüencial d'un sistema a una expressió algebraica sovint s'utilitza un quocient de dos polinomis depenents de la freqüència. És el mateix, diu, intentar ajustar a zero $E(s)$ que $A(s)E(s)$.

$$A(s)E(s) = B(s) - D(s)A(s) \simeq 0 \tag{4.7}$$

Per minimitzar aquest producte de polinomis on només es coneixen les dades $D(s)$, es construeix un sistema de matrius on Φ és el regressor, Θ el vector de paràmetres, i $\hat{Y}$ les dades.

$$\Phi\Theta = \hat{Y} \tag{4.8}$$

L'eina de MATLAB `invfreqs` resol aquest sistema d'equacions en el rang de freqüències de l'estudi, on `H` són les dades, `w` les freqüències, `nb` i `na` l'ordre del numerador i denominador respectivament, `Wt` un vector de pesos i `iter` el nombre de vegades que volem que es repeteixi el càlcul:

```
[B,A] = invfreqs(H,w,nb,na,Wt,iter)
```

Aquesta eina resol el sistema proporcionant dos polinomis que actuen com a numerador i denominador de la funció de transferència de l'ajust. A cada repetició els pesos es van reescalant amb les dades i el resultat del denominador, fent $(D \cdot A)^{-1}$, per tant, a l'afegir iteracions aquest resultat va sent més acurat. En la següent figura es mostra la convergència d'aquests pesos cap a un valor fix:

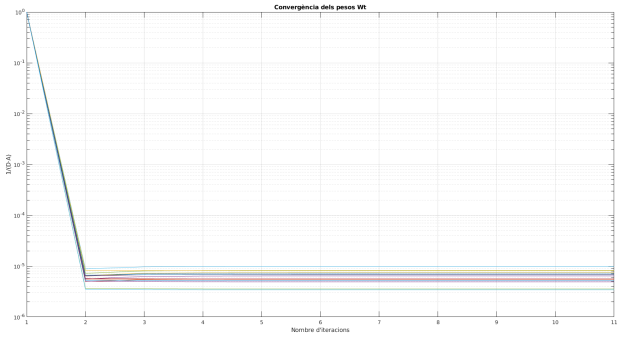


Figura 4.12: *Evolució dels pesos en l'assaig p*01_01*r*4*, en 16 iteracions*

En l'eina mencionada hi ha dos paràmetres lliures que es poden variar per realitzar un millor ajust: `nb` i `na`, els ordres del numerador i denominador de la funció de transferència, respectivament. Arribats a aquest punt, aquests valors poden anar-se variant fins a aconseguir l'ajust més idoni. (El que s'anomena fer cuina amb les dades). Sempre s'intentarà però, que per simplificar la síntesi del controlador, aquests valors siguin el més baixos possible.

Càlcul de l'índex d'Akaike [1]

L'índex d'Akaike (AIC[4]) és un indicador de la penalització que suposa incrementar l'ordre dels polinomis d'un model enfront a la semblança del propi model amb les dades experimentals. Aquest índex és menor com més ben triat estigui l'ordre `nb` i `na`, i s'expressa com

$$F = C \cdot \left(\frac{1 + \frac{d}{N}}{1 - \frac{d}{N}} \right) \tag{4.9}$$

on C és la funció de cost, d el nombre de paràmetres a estimar en el model, i N el nombre de dades de l'experiment (freqüències).
Per calcular-l'ho, un cop obtingut el model $G(j\omega)$, se'n calcula l'error fent:

$$Error(j\omega) = abs(D(j\omega) - G(j\omega)) \tag{4.10}$$

La funció de cost serà la suma dels quadrats d'aquest error, d valdrà $na + nb + 1$ i N valdrà 15. S'ha probat tres valors diferents de l'ordre per veure quin ajusta millor:

Ordres na i nb	**Índex d'Akaike**
2 i 2	4.16e7
3 i 3	4.60e6
4 i 4	4.17e7

Taula 4.2: *Índex d'Akaike per a diferents ordres en l'assaig* $p01\_01r4$

Per tant, en aquest cas s'ha utilitzat `na = 3` i `nb = 3`. La funció de transferència trobada en la última iteració, i el seu diagrama de Bode són:

$$FT_{final} = \frac{-131685.11 - 8586.46s + 63.16s^2 + 30.05s^3}{-2.28 + 35.56s - 5.10s^2 + s^3} = G_{Pitch,Vx} \tag{4.11}$$

[4]AIC: Akaike Information Criterion

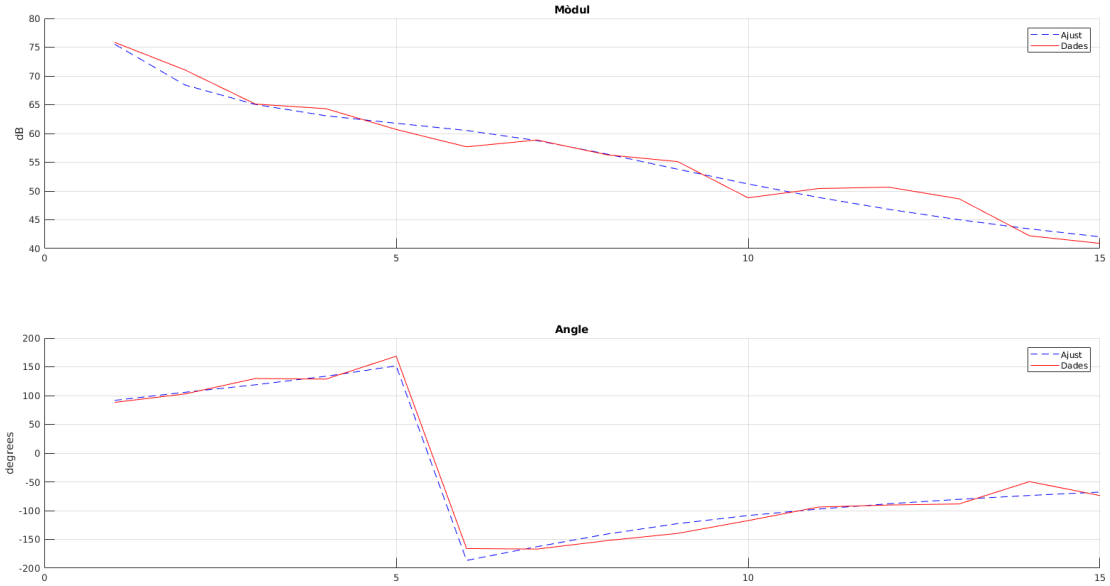


Figura 4.13: *Diagrama de Bode per l'ajust de les dades de l'assaig* $p01\_01r4$

4.5.3 Assaig de *Roll*

El nom de l'assaig per les dades d'entrada serà $r01\_01.mat$, i per les de sortida $r01\_01rX.mat$, on X representa el número d'intents que s'han enregistrat.
S'envia el senyal multisinus de la figura 4.4 , però es pren com a sortida la velocitat V_y.

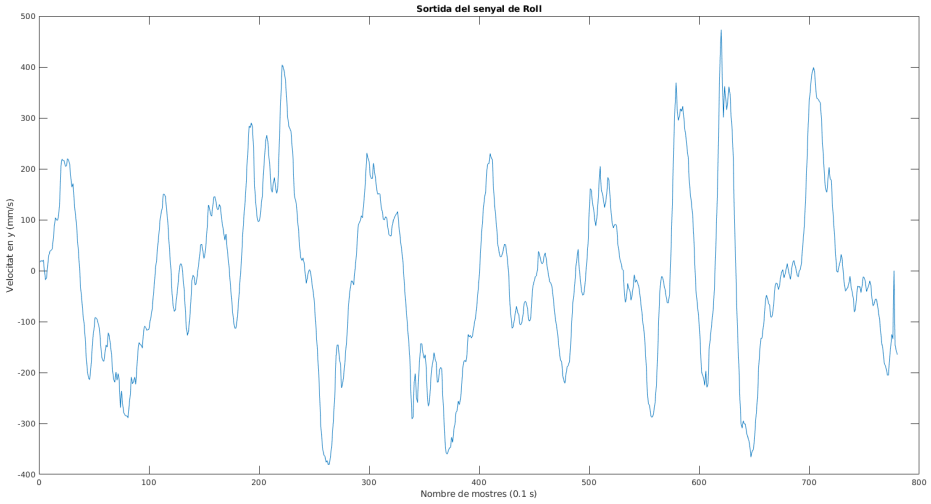


Figura 4.14: *Dades de sortida de l'assaig* $r01\_01r2$ *en domini temporal*

Igual que en el cas del *Pitch*, per identificar el sistema es passa a domini freqüencial:

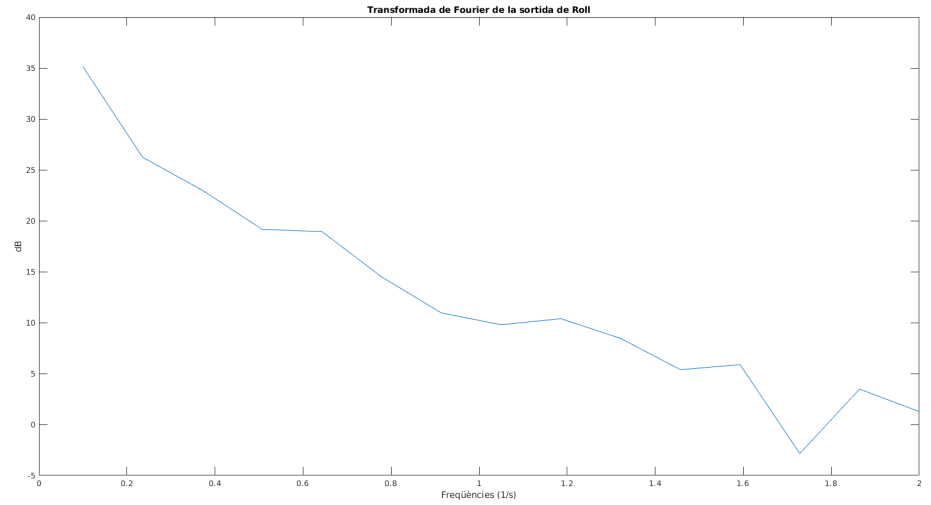


Figura 4.15: *Pics excitats de la sortida de l'assaig* $r01\_01r2$

Anàlisi dels resultats

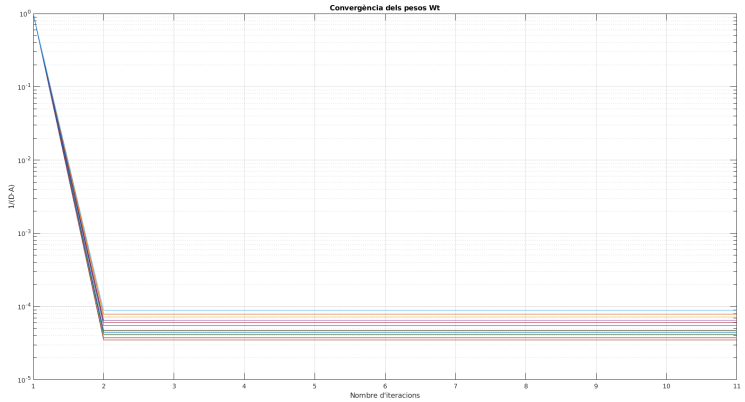


Figura 4.16: *Evolució dels pesos en l'assaig* $r01\_01r2$*, en 16 iteracions*

Igual que en el cas de *Pitch* s'ha calculat el valor de l'índex d'Akaike per esbrinar quin ordre seria més adient:

Ordres na i nb	Índex d'Akaike
2 i 2	9.30e6
3 i 3	3.85e7

Taula 4.3: *Índex d'Akaike per a diferents ordres en l'assaig* $r01\_01r2$

Així doncs, triant $nb = 2$ i $na = 2$, la funció de transferència trobada en la última iteració, i el seu diagrama de Bode:

$$FT_{final} = \frac{19715.07 + 1444.48s + 212.20s^2}{2.72 - 3.05s + s^2} = G_{Roll,Vy} \tag{4.12}$$

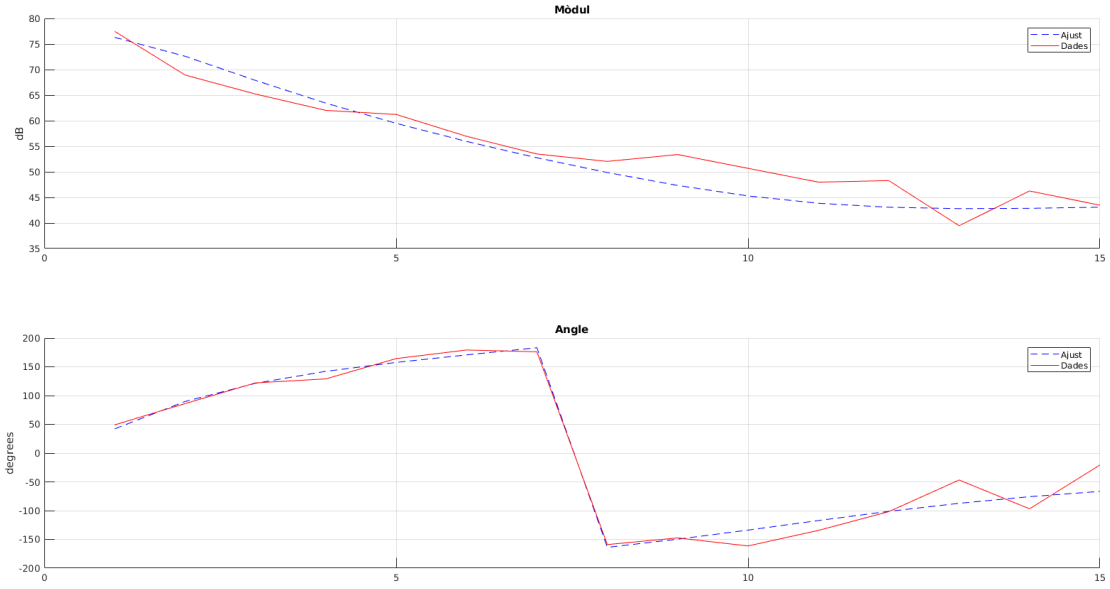


Figura 4.17: *Diagrama de Bode per l'ajust de les dades de l'assaig* $r01\_01r2$

4.5.4 Assaig de *Yaw*

El nom de l'assaig per les dades d'entrada serà $y01\_01.mat$, i per les de sortida $r01\_01rX.mat$, on X representa el número d'intents que s'han enregistrat.
S'envia el senyal multisinus de la figura 4.4 , però es pren com a sortida l'angle mesurat de *Yaw*, ψ, ja que no es proporciona la velocitat en z.

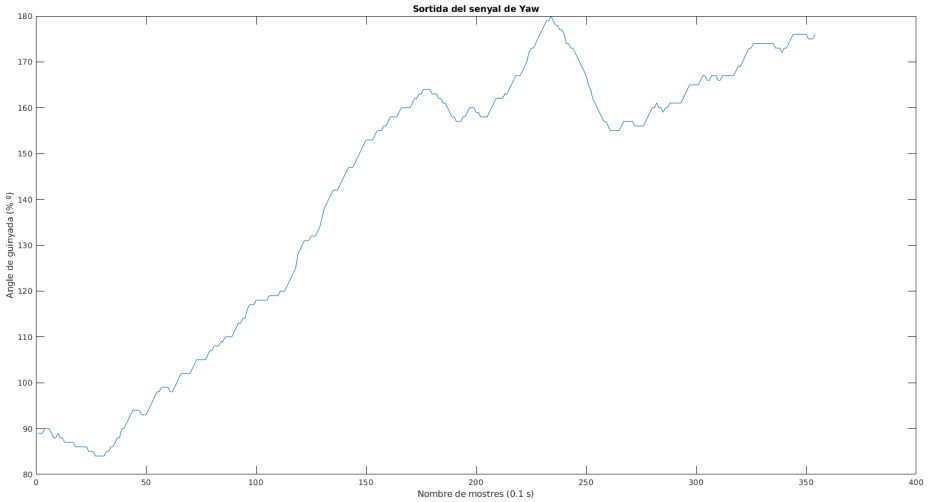


Figura 4.18: *Dades de sortida de l'assaig* $y01\_01r2$ *en domini temporal*

Igual que en el cas del *Pitch*, per identificar el sistema es passa a domini freqüencial:

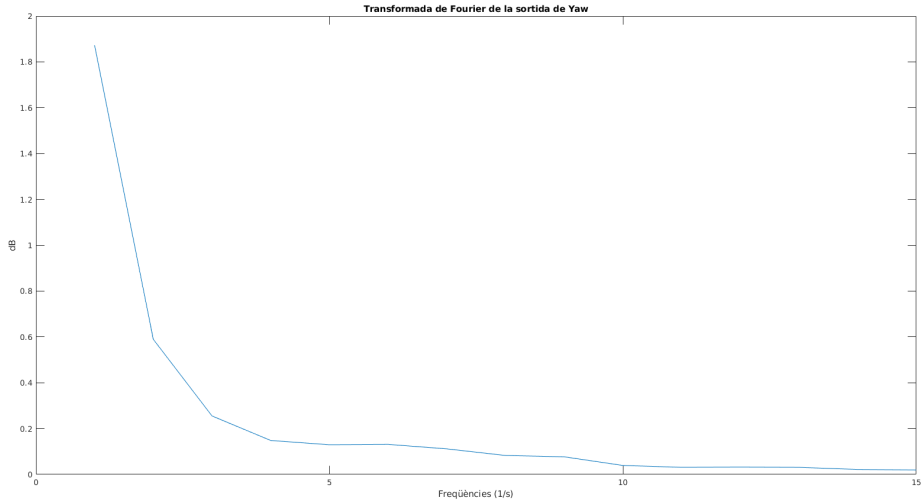


Figura 4.19: *Pics excitats de la sortida de l'assaig* $y01\_01r2$

Anàlisi dels resultats

Igual que en el cas de *Pitch* s'ha calculat el valor de l'índex d'Akaike per esbrinar quin ordre seria més adient:

Ordres na i nb	Índex d'Akaike
2 i 1	4.68e4
2 i 2	2.07e4
3 i 2	1.27e7

Taula 4.4: *Índex d'Akaike per a diferents ordres en l'assaig* $y01\_01r2$

Així doncs, triant $nb = 2$ i $na = 2$, la funció de transferència trobada en la última iteració, i el seu diagrama de Bode:

$$FT_{final} = \frac{-128.08 + 18.48s + 2.30s^2}{0.97 - 0.55s + s^2} = G_{Yaw,\psi} \tag{4.13}$$

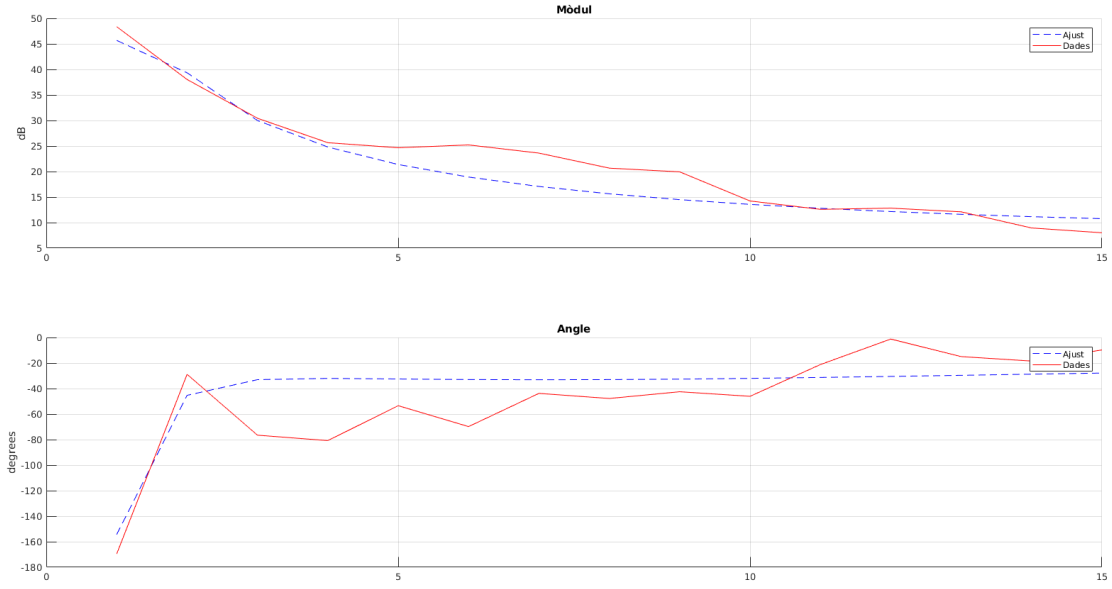


Figura 4.20: *Diagrama de Bode per l'ajust de les dades de l'assaig* $y01\_01r2$

Capítol 5

Disseny d'un controlador sobre velocitat lineal

5.1 Quin és l'objectiu de l'obtenció d'un controlador?

El quadrotor és un sistema inherentment inestable com es veurà a continuació, però això implica que posseeix una gran maniobrabilitat. No obstant, per aconseguir el moviment desitjat requereix un control adequat.
Si es desitja controlar el sistema de la figura 4.2 s'ha de tancar el llaç afegint una funció anomenada C_1 que actui com a controlador

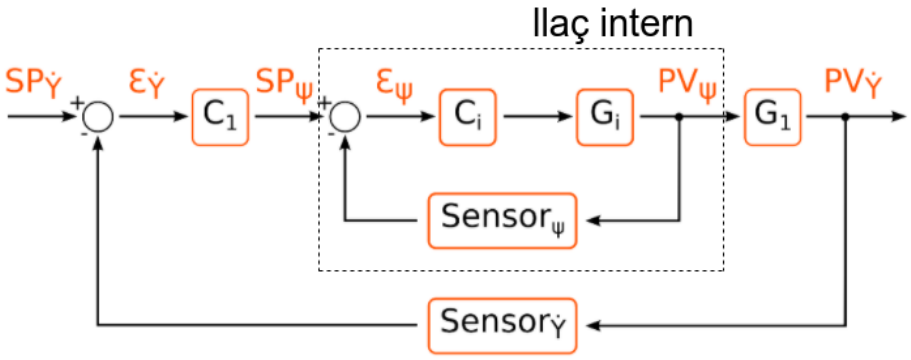


Figura 5.1: *Imatge extreta de [9]*

5.2 Disseny d'un controlador pel *Pitch*

Primerament, s'observa que en simular el sistema sense controlador, amb un graó resulta clarament inestable:

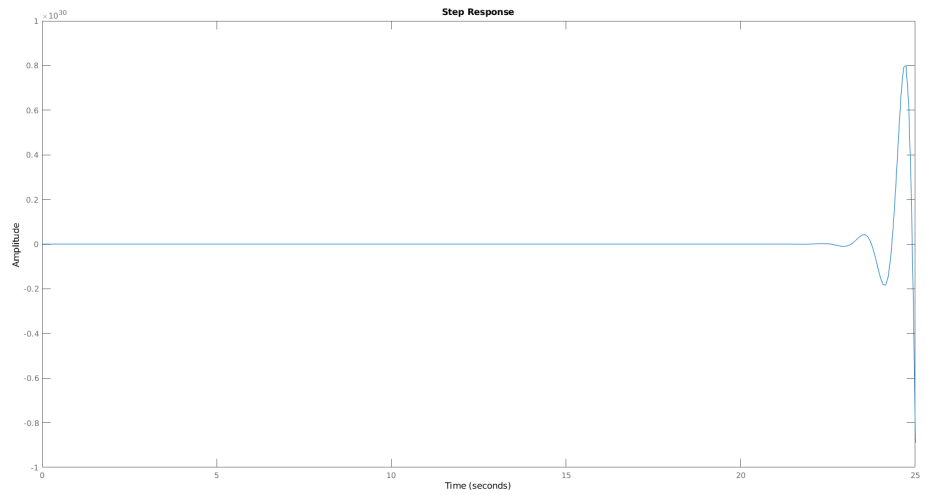


Figura 5.2: *Resposta a un graó per l'ajust de les dades de l'assaig* $p01\_01r4$

El mètode de síntesi directa planteja la funció de transferència d'un llaç tancat sobre el sistema real del *Dron*, per convertir-ho en un sistema desitjat. Si s'agafa el llaç intern i el model G_1 com a un nou model G, i se li aplica un controlador C l'expressió queda:

$$\frac{C \cdot G}{1 + C \cdot G} = G_{desitjat} = \frac{K}{\tau s + 1} \tag{5.1}$$

En aquest cas, es vol aconseguir un sistema de primer ordre tal que arribi al valor d'entrada exacte (per tant el guany ha de ser $K = 1$) en el menor temps possible (valor de τ petit). Aillant-lo de l'expressió anterior:

$$C = \frac{1}{G} \cdot \frac{G_{desitjat}}{1 - G_{desitjat}} = \frac{A}{B} \cdot \frac{K}{\tau s + 1 - K} = \frac{1 + a_1 s + a_2 s^2}{b_0 + b_1 s + b_2 s^2} \cdot \frac{1}{\tau s} \tag{5.2}$$

Passat a MATLAB, es simplifica l'expressió fent

```
NumControl=conv(A,1)
DenControl=conv(B,[tau  0])
```

Així s'obté un primer controlador. Per a què s'ajusti més a la realitat es pot ajustar en funció de la distribució de zeros i pols del sistema real, és a dir, se simula el model amb la FT trobada i s'aplica el `pzmap(model)`:

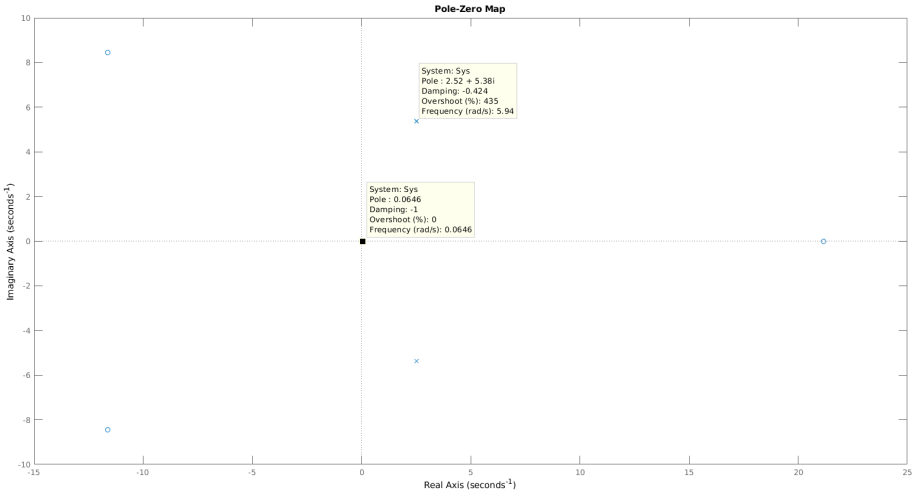


Figura 5.3: *Pols i zeros de les dades de l'assaig* $p01\_01r4$

Es veuen tres pols inestables (reals positius), dos a $(2.52, j5.38)$ i un a 0.06, i dos zeros conjugats.Per tant, al controlador ja calculat C, com que és difícil de manipular degut al seu elevat ordre, se li pot aproximar un altre controlador que proporcioni una resposta semblant en el rang desitjat. Al final el que es vol aconseguir és que tingui la forma d'un PID [1] tal que:

$$C_{aprox}(s) = \frac{a_2 s^2 + a_1 s + a_0}{s} \tag{5.3}$$

Una opció és la d'aplicar el criteri de Routh al numerador del controlador $(-2.28 + 35.56s - 5.10s^2 + s^3)$ i, una vegada muntada la matriu, agafar la reduïda a partir de l'ordre 2, que és el que es vol aconseguir.

Segons l'article [13] es pot aplicar una reducció d'ordre a un model seguint el criteri d'estabilitat de Routh. Si es construeix la taula dels coeficients, i es trunca a la fila de l'ordre desitjat, es pot aproximar el comportament d'un nou model d'ordre inferior al comportament del model original. Per exemple del numerador $-2.28 + 35.56s - 5.10s^2 + s^3$ es passaria a $-2.28 + 35.56s - 5.10s^2$.
Si la baixada d'ordre no és molt dràstica, el nou model reduït preserva el comportament freqüencial a baixes freqüències del sistema original. Com que les maniobres que se li manaràn al *Dron* no són ràpides, és una bona manera de reduir l'ordre del controlador mantenint prestacions dinàmiques de baixes freqüències.

Així, es crea un numerador que sigui $-2.28 + 35.56s - 5.10s^2$ i el denominador s, i només queda ajustar-li el guany. En la següent figura es presenten els dos possibles controladors en un diagrama de Bode:

[1] PID: Controlador Proporcional Integral Derivatiu

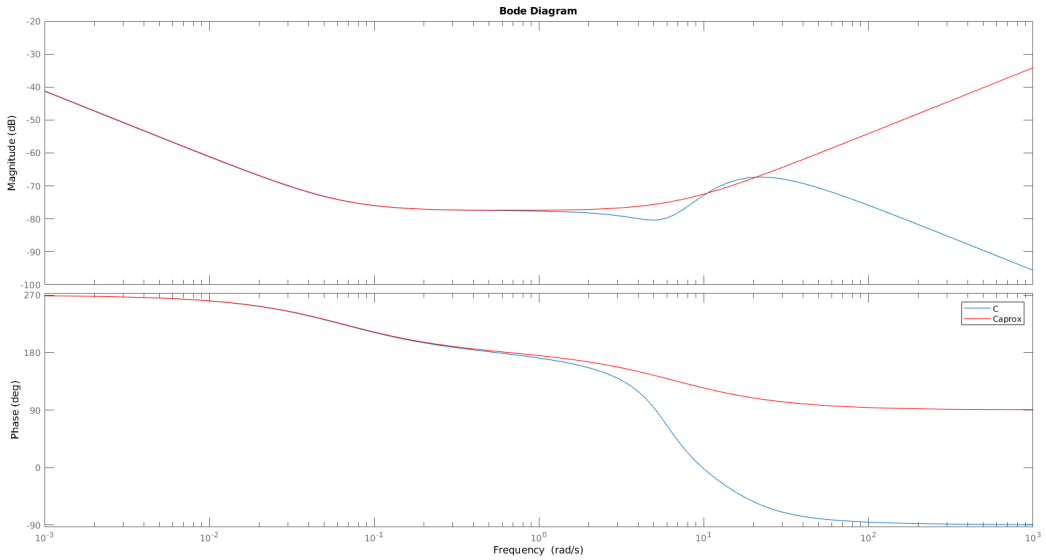


Figura 5.4: *Magnitud i angle del controlador de les dades de l'assaig p*01_01*r*4

Es pot veure que en les freqüències en les que treballa el Parrot (entre 0.1 i 2 Hz), els dos controladors actuen pràcticament igual, per tant, l'escollit hauria de ser el més simple possible:

$$C_{aprox} = \frac{0.88 \cdot 10^{-5}s^2 - 13.5 \cdot 10^{-5}s + 1.96 \cdot 10^{-5}}{s} = C_{Pitch,Vx} \tag{5.4}$$

No obstant, al comprovar els resultats d'aquest apartat realitzant un model de SIMULINK, el sistema no respon de manera estable, ni amb el controlador sintetitzat ni amb l'aproximat. Veient la distribució de pols i zeros del sistema ja s'intueix que el pol situat a x=21.2 fa que el sistema sigui inestable. Arribats a aquest punt, es pot moure aquest pol de 21.2 a -21.2, que canviarà la fase però no el mòdul.
Així, es grafica la resposta simulada:

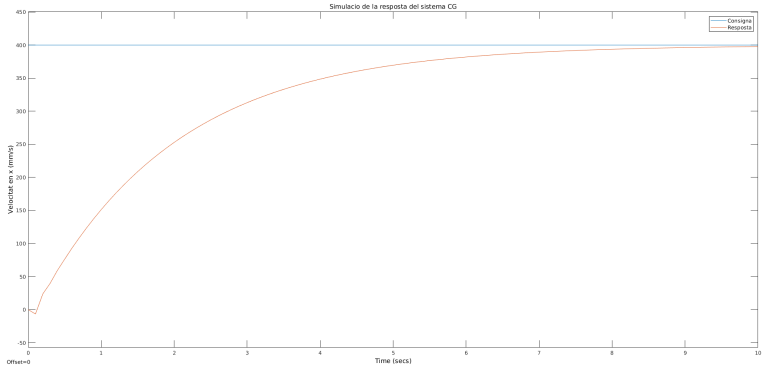


Figura 5.5: *Resposta simulada amb el controlador de síntesi directa modificat, per l'angle de Pitch*

Aquest controlador, a part de ser molt lent, no és fàcil de manipular, i el controlador que s'havia

aproximat no estabilitza bé el sistema. No obstant, es pot recórrer a una segona opció: com que el règim que interessa és el de freqüències baixes, s'hi pot aproximar un PI [2] en aquestes freqüències.
Es plantegen unes constants de, per exemple, $K_p = 0.0005$ i $K_i = 0.0001 seg^{-1}$ i s'analitza el Bode d'aquest controlador enfront de l'anterior:

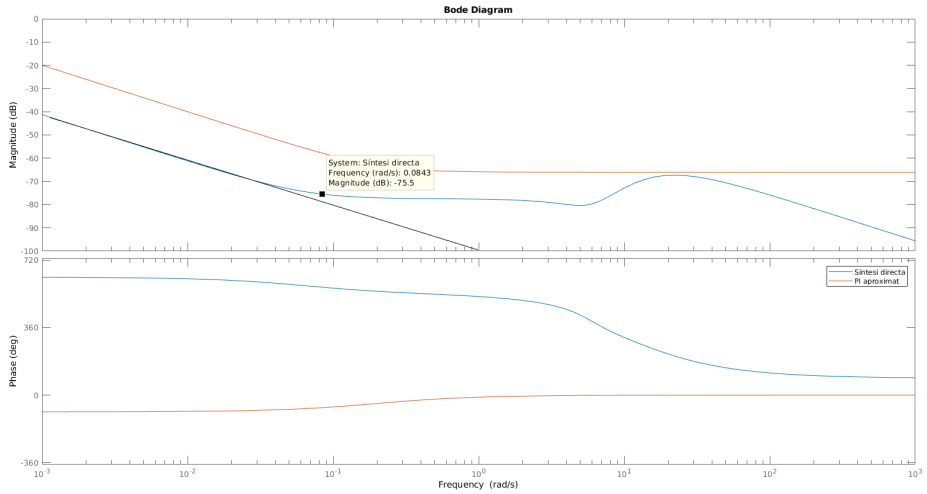


Figura 5.6: *Comparativa de controladors de Pitch*

El punt on el controlador calculat per síntesi directa canvia de pendent (el "colze" de la gràfica) correspon a $\frac{K_p}{K_i}$ i, si seguim una recta del mateix pendent que les baixes freqüències fins al valor $w = 1 rad/s$, la intersecció dóna el valor de K_i. Així, es determina un nou controlador de paràmetres

$$K_p = 0.000119 \qquad K_i = 0.00001$$

i es compara amb el de la síntesi directa:

[2] Controlador Proporcional Integrador

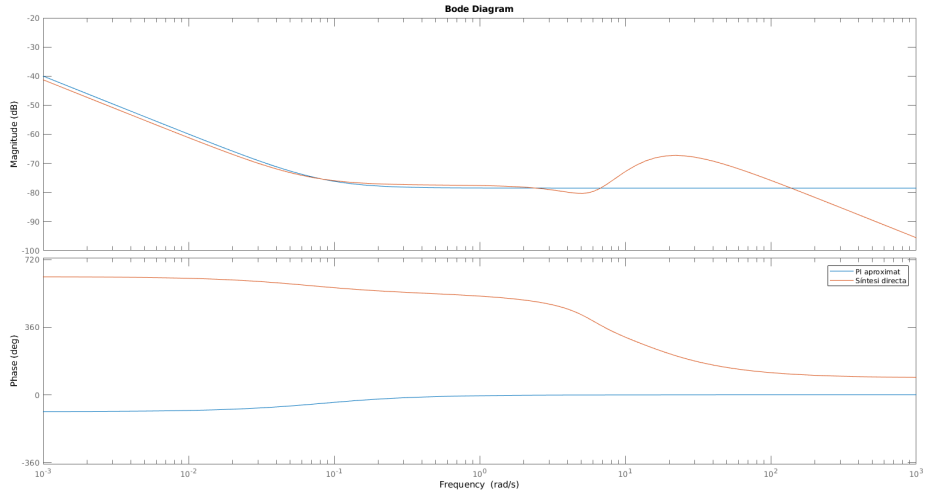


Figura 5.7: *Comparativa de controladors de Pitch*

S'ha comprovat l'efectivitat d'aquest controlador davant d'un graó excitat a l'aeronau real:

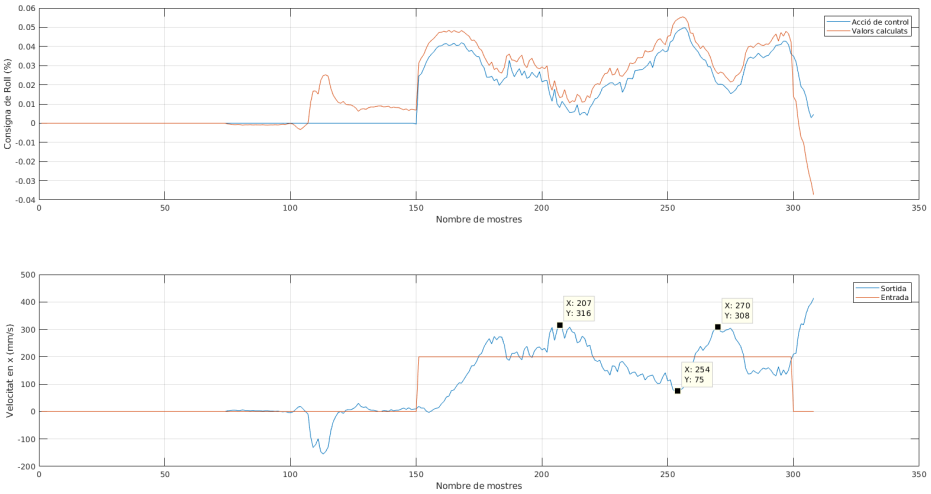


Figura 5.8: *Resposta real de l'aeronau amb el PI aproximat, per l'angle de Pitch*

5.3 Disseny d'un controlador pel *Roll*

Primer s'analitza la resposta del primer ajust a un graó. Igual que en el cas anterior, al realitzar l'assaig amb un senyal ric en freqüències, la identificació proporciona un sistema inestable:

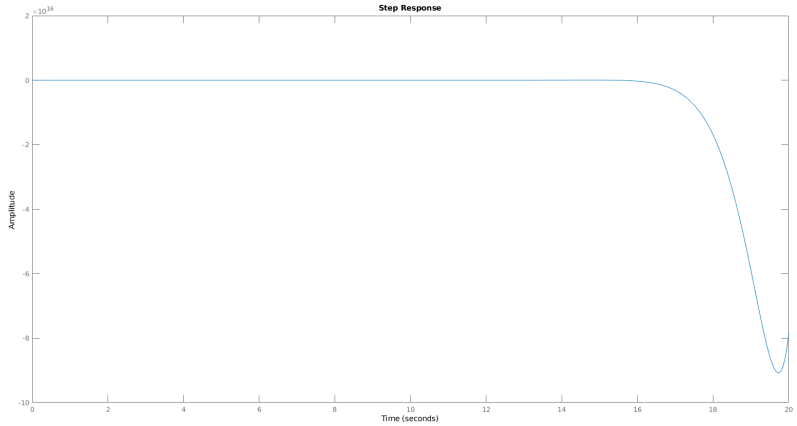


Figura 5.9: *Resposta a un graó per l'ajust de les dades de l'assaig* $r01\_01r2$

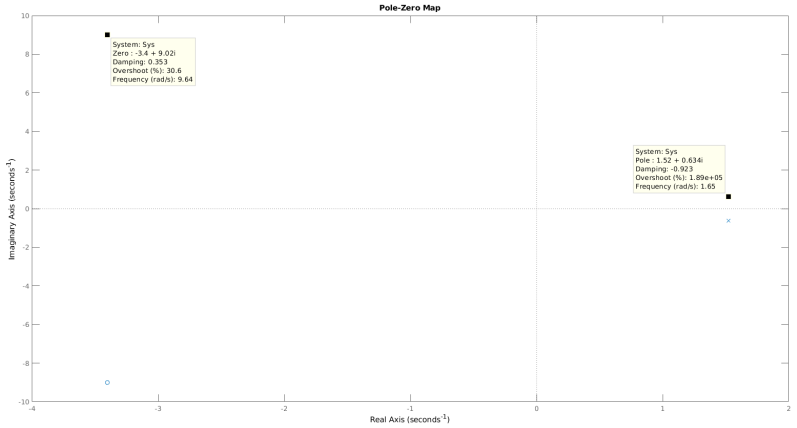


Figura 5.10: *Pols i zeros de les dades de l'assaig* $r01\_01r2$

Es pot veure que hi ha dos pols amb part real positiva a $(1.52, j0.634)$ que es tindran en compte al numerador del controlador, i s'aproximarà un de nou que respongui com un proporcional integrador derivatiu. Utilitzant MATLAB, es crida l'eina `poly` que proporciona els coeficients d'un polinomi amb les arrels que se li indiquin.

$$C_{aprox}(s) = \frac{Num(s)}{s} \tag{5.5}$$

Així, en la següent figura es presenten els dos possibles controladors en un diagrama de Bode:

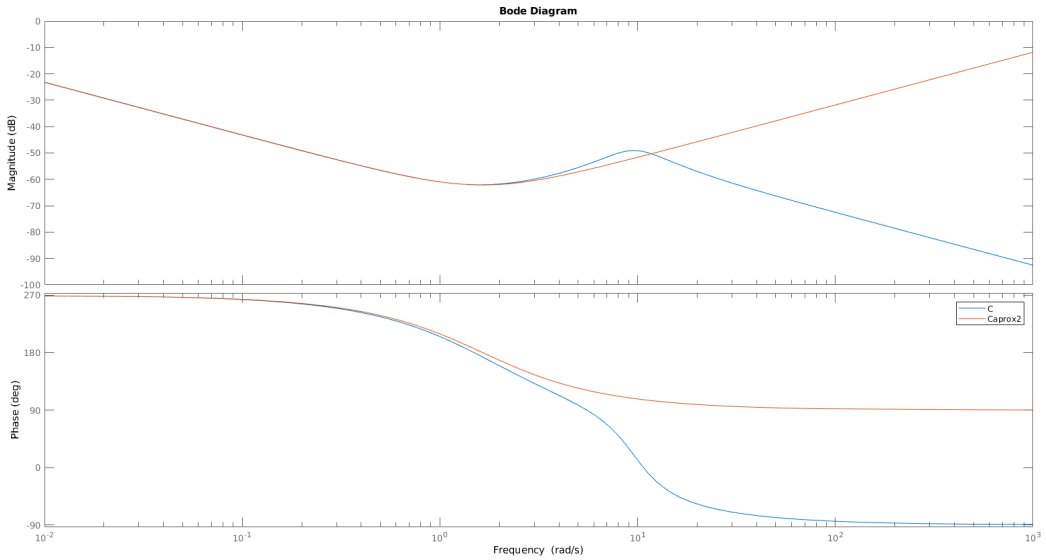


Figura 5.11: *Magnitud i angle del controlador de les dades de l'assaig* $r01\_01r2$

Finalment, per comprovar tots aquests resultats es realitza un model de SIMULINK amb un llaç tancat on hi aparegui el model identificat de l'angle de *Pitch* i el controlador sintetitzat. Es grafica per tant la resposta simulada:

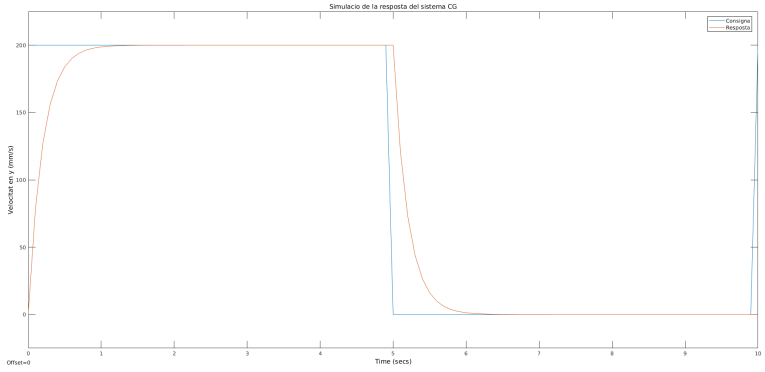


Figura 5.12: *Resposta simulada per l'angle de Roll*

Es pot veure altre cop en la figura 5.11 que en les freqüències en les que treballa el Parrot (entre 0.1 i 2 Hz), els dos actuen pràcticament igual. L'escollit hauria de ser el més simple possible, un PID :

$$C_{aprox} = \frac{2.57 \cdot 10^{-4} s^2 - 7.81 \cdot 10^{-4} s + 6.97 \cdot 10^{-4}}{s} = G_{Roll,Vy} \tag{5.6}$$

No obstant, al simular-ne la resposta amb el PID no queda estable. En la següent secció es provaran mètodes alternatius.

5.4 Disseny d'un controlador pel *Yaw*

Primer s'analitza la resposta del primer ajust a un graó. Igual que en els casos anteriors, al realitzar l'assaig amb un senyal ric en freqüències, la identificació proporciona un sistema inestable:

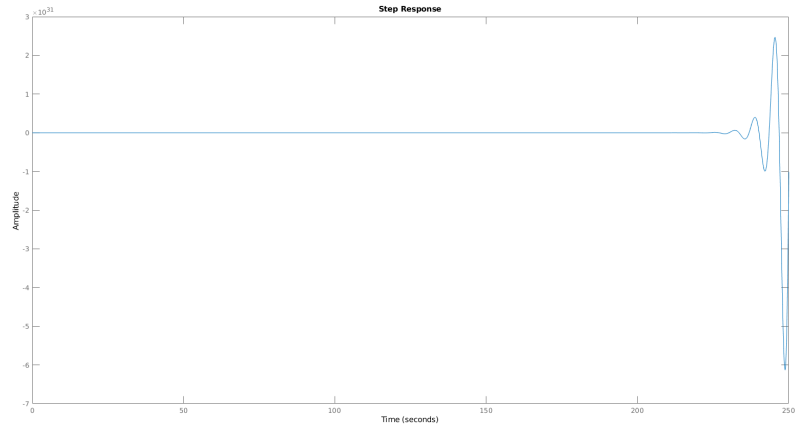


Figura 5.13: *Resposta a un graó per l'ajust de les dades de l'assaig* $y01\_01r2$

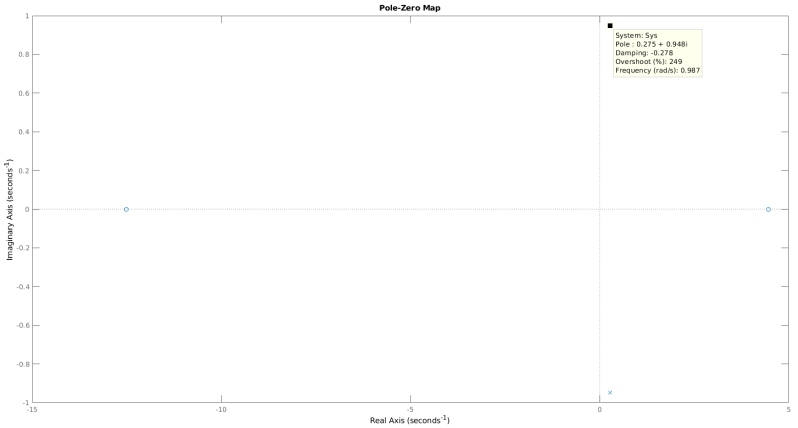


Figura 5.14: *Pols i zeros de les dades de l'assaig* $y01\_01r2$

Es pot veure que hi ha dos pols amb part real positiva a $(0.275, j0.948)$ que es tindran en compte al numerador del controlador, i s'aproximarà un de nou que respongui com un proporcional integrador derivatiu. Utilitzant MATLAB, es crida l'eina `poly` que proporciona els coeficients d'un polinomi amb les arrels que se li indiquin.

$$C_{aprox}(s) = \frac{Num(s)}{s} \tag{5.7}$$

Així, en la següent figura es presenten els dos possibles controladors en un diagrama de Bode:

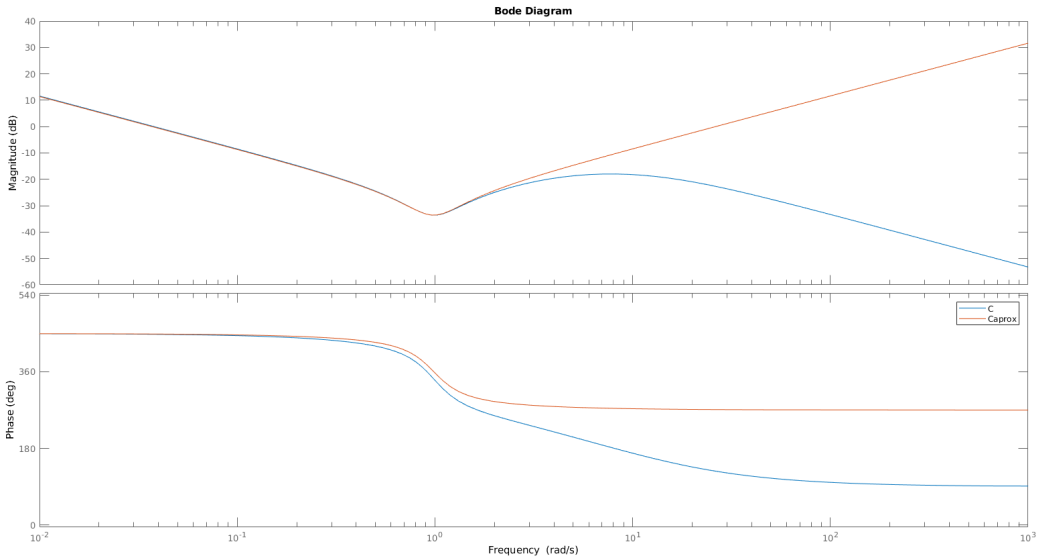


Figura 5.15: *Magnitud i angle del controlador de les dades de l'assaig* $y01\_01r2$

Finalment, per comprovar tots aquests resultats es realitza un model de SIMULINK amb un llaç tancat on hi aparegui el model identificat de l'angle de *Pitch* i el controlador sintetitzat. Es grafica per tant la resposta simulada:

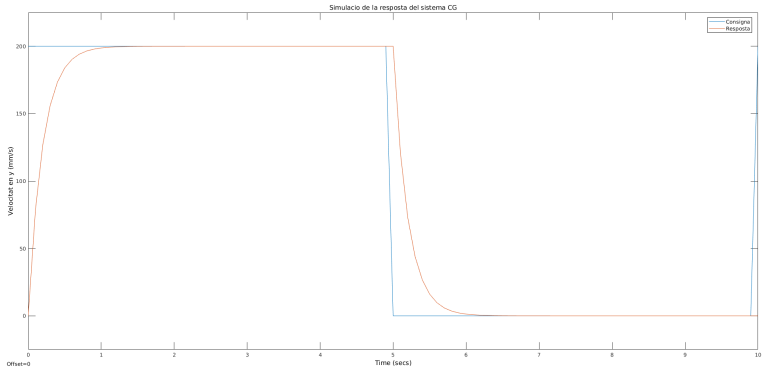


Figura 5.16: *Resposta simulada per l'angle de Yaw*

Es pot veure altre cop en la figura 5.15 que en les freqüències en les que treballa el Parrot (entre 0.1 i 2 Hz), els dos actuen pràcticament igual. L'escollit hauria de ser el més simple possible, un PID :

$$C_{aprox} = \frac{-0.0381s^2 + 0.021s - 0.0371}{s} = G_{Yaw,\psi} \tag{5.8}$$

Com que no resulta estable, es provaran altres mètodes.

5.5 Resintonització de controladors

Tant la resposta de *Pitch* com la de *Roll* amb controladors segueixen sent inestables. Això no permetrà tenir un bon control de la dinàmica del *Dron*, per tant es proposen camins alternatius:

- Probar valors semialeatoris en la màquina real
- Considerar acoblament de les variables
- Identificació robusta

5.5.1 Resintonització de Chidambara

Veient les respostes obtingudes es proposa, amb assessorament del director, realitzar resintonitzacions dels paràmetres dels controladors.

Resintonització de *Pitch*

És fàcil veure a la figura 5.8 que la sortida segueix prou bé el graó però amb fortes oscil·lacions, per això s'ha considerat utilitzar la resintonització de Chidambara, amb el propòsit d'evitar-les.

Per una resposta genèrica com la de la figura següent, es poden caracteritzar les elongacions i el període d'oscil·lació per calcular nous controladors.
El procediment és el següent:

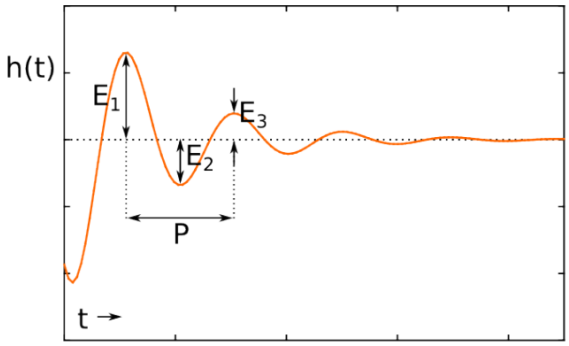


Figura 5.17: *Paràmetres a obtenir per la resintonització*

$$\Delta = -\frac{1}{2\pi}\ln\left(\frac{E_3}{E_1}\right) \tag{5.9}$$

$$\Delta_{des} = -\frac{1}{2\pi}\ln\left(\frac{E_3}{E_1}\right)_{des} \tag{5.10}$$

$$K_{p_{nova}} = \frac{1+4.5454\Delta}{1+4.5454\Delta_{des}}K_p \tag{5.11}$$

$$T_{i_{nou}} = \frac{P}{1.2\sqrt{1+\Delta^2}} \tag{5.12}$$

$$K_{i_{nova}} = \frac{K_{p_{nova}}}{T_{i_{nou}}} \tag{5.13}$$

Així, es troben els valors del controlador PI definitiu

$$K_p = 0.000057825 \quad K_i = 0.0000011015\ seg^{-1} \quad K_d = 0\ seg \tag{5.14}$$

i es proven a la màquina.

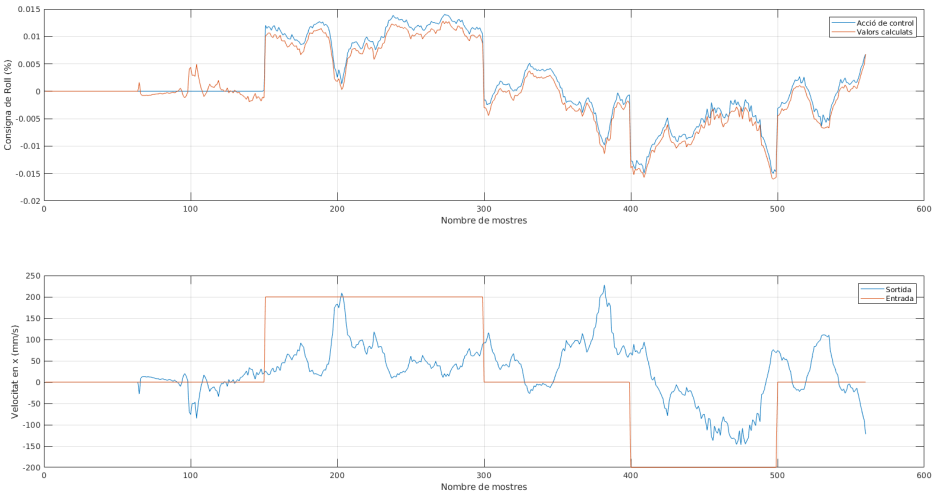


Figura 5.18: *Resintonització de Chidambara per Pitch*

Els valors d'aquestes constants són molt baixos, per això el controlador és poc agressiu i no s'assoleix en cap cas la consigna. Per aconseguir el comportament desitjat, per tant, s'haurà de recórrer a altres mètodes.

Resintonització de *Roll*

Es proven uns valors de partida per *Roll* iguals, tals que

$$K_p = 0.0005 \quad K_i = 0.00001\ seg^{-1} \quad K_d = 0\ seg \tag{5.15}$$

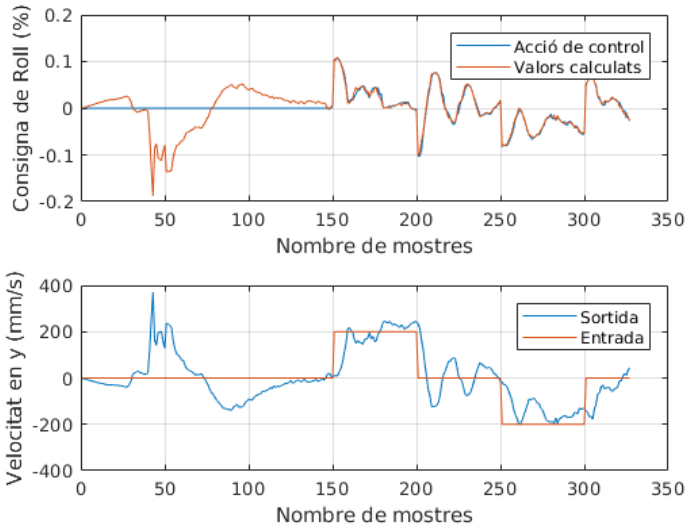


Figura 5.19: *Test de controlador de Roll*

És fàcil veure que la sortida segueix prou bé el graó però amb fortes oscil·lacions, per això s'ha considerat utilitzar la resintonització de Chidambara, amb el propòsit d'evitar-les. Així, es troben els valors del controlador PI definitiu

$$K_p = 0.000336 \quad K_i = 0.0000192\ seg^{-1} \quad K_d = 0\ seg \tag{5.16}$$

Reproduint el mateix assaig que a la situació anterior de la figura 5.19 les oscil·lacions de la resposta de velocitat quan se li envia un senyal de pols disminueixen significativament.

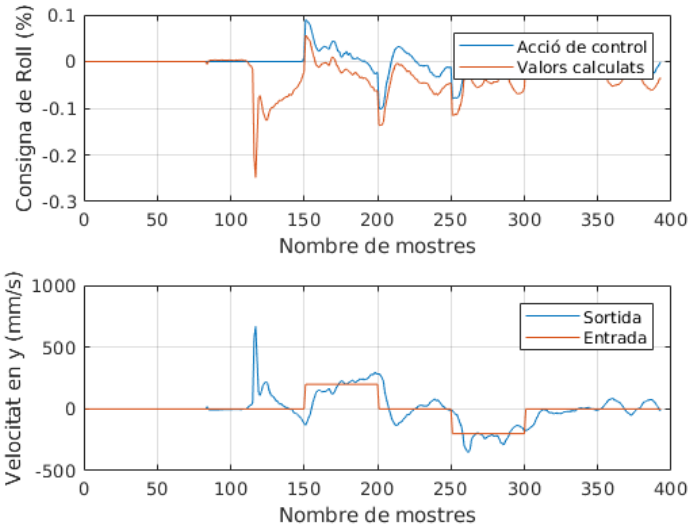


Figura 5.20: *Resintonització de Chidambara pel Roll*

5.5.2 Desacoblament de les variables: Models creuats

S'ha detectat que en enviar una consigna de, per exemple, *Pitch* al *Dron*, es rep una resposta en la velocitat lineal en x com era d'esperar, però també en y. De la mateixa manera, quan s'envien consignes de *Roll* es detecta moviment en y i en x. Els controladors que s'han trobat fins ara corresponen als dos models de *Pitch* i *Roll* però desacoblats. Si es vol captar la dinàmica real del vehicle s'ha de considerar que les variables estan acoblades, i dissenyar controladors específics per desacoblar-les.

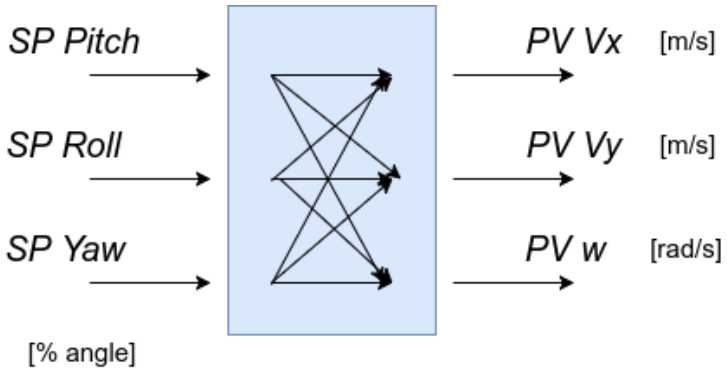


Figura 5.21: *Esquema d'entrades i sortides acoblades*

En la següent imatge s'aprecia el sistema acoblat, amb dues funcions de transferència addicionals a les $G_{11} = G_{Pitch,Vx}$ i $G_{22} = G_{Roll,Vy}$ que ja es tenien: $G_{12} = G_{Roll,Vx}$ i $G_{21} = G_{Pitch,Vy}$.

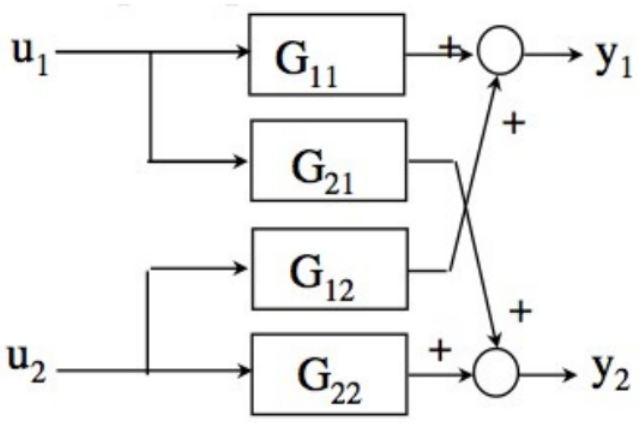


Figura 5.22: *Variables acoblades*

Per desacoblar-les s'hauria d'implementar l'esquema següent:
Per trobar primer les funcions de transferència de cada model creuat es procedeix de la mateixa manera que en els casos anteriors, però creuant els *outputs*:

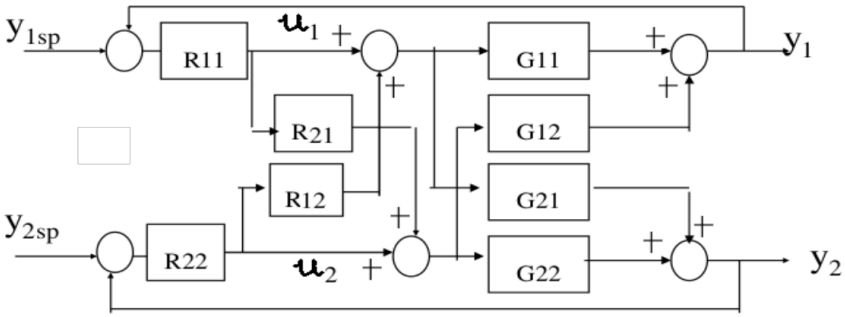


Figura 5.23: *Controladors acoblats*

Entrada de *Pitch* i sortida de V_y

S'identifica, igual que en el capítol anterior, tenint en compte l'índex d'Akaike per ajustar millor el model, i s'escull un ordre $nb = 2$, $na = 2$.

$$FT_{final} = \frac{1587.22 + 55.59s + 32.61s^2}{4.06 - 0.08s + s^2} = G_{Pitch,Vy} \tag{5.17}$$

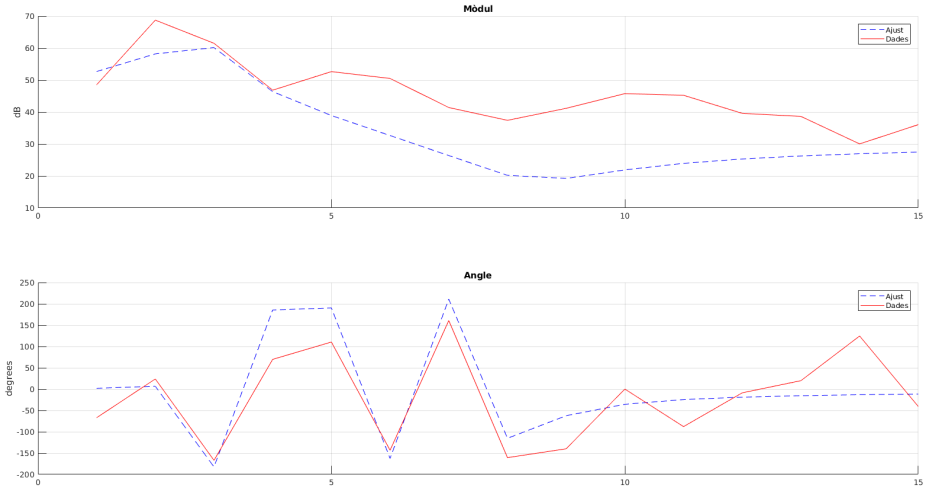


Figura 5.24: *Diagrama de Bode per l'ajust de les dades de l'assaig p*01_01*r*4

Entrada de *Roll* i sortida de V_x

S'identifica, igual que en el capítol anterior, tenint en compte l'índex d'Akaike per ajustar millor el model, i s'escull un ordre $nb = 3$, $na = 3$.

$$FT_{final} = \frac{-3927.79 - 696.66s - 426.00s^2 - 34.50s^3}{-0.34 + 9.05s + -0.19s^2 + s^3} = G_{Roll,Vx} \tag{5.18}$$

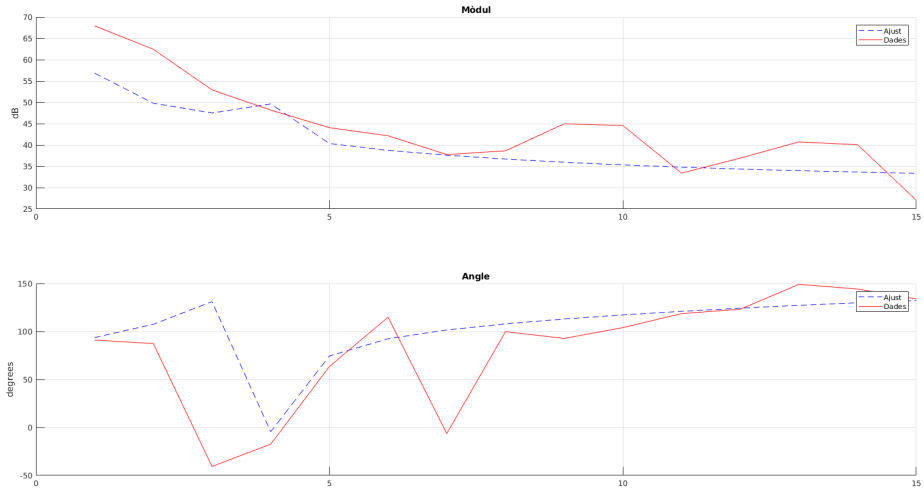


Figura 5.25: *Diagrama de Bode per l'ajust de les dades de l'assaig* $r01\_01r2$

Un cop identificats els models creuats es poden comparar amb els diagrames de Bodes de les funcions directes, $G_{Pitch,Vx}$ i $G_{Roll,Vy}$. Analitzant el mòdul es pot calcular com de gran és el guany d'un respecte l'altre. S'ha vist, per tant, que el guany dels models creuats és més d'un 30% més petit que el dels models no creuats per a la mateixa variable (V_x, V_y). Finalment, i com s'ha dit a l'estudi previ [9], es pot considerar que no es manifesta aquest acoblament dels models.

Capítol 6

Identificació robusta

6.1 Algorisme de Bhattacharyya

Amb l'objectiu d'obtenir una identificació més robusta que la realitzada a través d'un sol assaig amb el *Dron*, s'utilitza l'algorisme de Bhattacharyya [**?**] que proporciona un model on els coeficients són intervalars, i embolcallen tot el comportament freqüencial dels tests. Aquest sistema permet predir fins al pitjor comportament del sistema, i determinar els marges d'estabilitat. Es defineix com

$$\mathbf{G}(s) = \frac{\hat{n_0} + \hat{n_1}s + \hat{n_2}s^2 + ... + \hat{n_n}s^n}{1 + \hat{d_1}s + \hat{d_2}s^2 + ... + \hat{d_n}s^n} \tag{6.1}$$

on apareixen coeficients $\hat{n}_i$ i $\hat{d}_i$ que no tindran un únic valor, sino infinits. Així, s'identificaran infinits models per a aquesta planta. Per trobar aquests valors es necessita:

$$\mathbf{G}(s) = \{G(s) : \hat{n_i} \in \left[n_i + w_{n_i}\epsilon^-_{n_i}, n_i + w_{n_i}\epsilon^+_{n_i}\right], \hat{d_i} \in \left[d_i + w_{d_i}\epsilon^-_{d_i}, d_i + w_{d_i}\epsilon^+_{d_i}\right], A\ i\} \tag{6.2}$$

L'algorisme consisteix en identificar primer els pesos $w = [w_{d_0}...w_{d_n} \quad w_{n_0}...w_{n_n}]$, i després els termes $\epsilon^+ = [\epsilon^+_{d_0}...\epsilon^+_{d_n} \quad \epsilon^+_{n_0}...\epsilon^+_{n_n}]$ i $\epsilon^- = [\epsilon^-_{d_0}...\epsilon^-_{d_n} \quad \epsilon^-_{n_0}...\epsilon^-_{n_n}]$, complint tres restriccions importants:

1. Les dades experimentals han d'estar contingudes dins el model $\mathbf{G}(j\omega_i)$
2. $||\epsilon^{\pm}||$ mínims possibles
3. Els pesos w s'han d'escollir de tal manera que la resposta freqüencial intervalar s'ajusti el màxim possible a la resposta de les dades

Un pes w_{n_i} representa la sensitivitat d'un coeficient del model nominal respecte la variació de les dades experimentals, així doncs es calcula com la mitjana de les sensitivitats per a una freqüència concreta. Primer, es creen l models $\mathbf{G}^l(s)$ (tants com dades freqüencials) idèntics al nominal excepte per un dels valors de partida, calculat amb el model. Així, es crea una matriu tal que:

$$\begin{array}{cccccc} |n_0 - n_0^1| & ... & |n_n - n_n^1| & |d_0 - d_0^1| & ... & |d_n - d_n^1| \\ |n_0 - n_0^2| & ... & |n_n - n_n^2| & |d_0 - d_0^2| & ... & |d_n - d_n^2| \\ & & . & & & \\ |n_0 - n_0^N| & ... & |n_n - n_n^N| & |d_0 - d_0^N| & ... & |d_n - d_n^N| \end{array}$$

i es calculen els pesos:

$$w = \frac{1}{N}[\sum_{i=1}^{N} |n_0 - n_0^l| \ \ ... \ \ |d_n - d_n^l|] = w_{n_0}...w_{n_n}w_{d_0}...w_{d_n} \tag{6.3}$$

A continuació cal recuperar la forma polar de les dades experimentals $D(j\omega_i) = \alpha_i + j\beta_i$, per determinar els paràmetres $\epsilon^{\pm}$, fent servir l'expressió

$$A(\omega_i, \alpha_i, \beta_i)W\epsilon^i = B(\omega_i, \alpha_i, \beta_i) \tag{6.4}$$

on $A(\omega_i, \alpha_i, \beta_i) =$

$$\begin{Bmatrix} \alpha_i & -\beta_i\omega_i & -\alpha_i\omega_i^2 & ... & -1 & 0 & \omega_i^2 & ... \\ \beta_i & \alpha_i\omega_i & -\beta_i\omega_i^2 & ... & 0 & \omega_i & 0 & ... \end{Bmatrix}$$

i $B(\omega_i, \alpha_i, \beta_i) =$

$$\begin{Bmatrix} -\alpha_i(d_0 - \omega_i^2 d_2 + ...) + \beta_i(\omega_i d_1 + ...) + (n_0 - \omega_i^2 n_2 + ...) \\ -\beta_i(d_0 - \omega_i^2 d_2 + ...) - \alpha_i(\omega_i d_1 + ...) + (\omega_i n_1 + ...) \end{Bmatrix}$$

i $W = Diag[w_{d_0}, ..., w_{n_0}, ...]$ i $\epsilon^i = [\epsilon^i_{d_0}...\epsilon^i_{d_n}\epsilon^i_{n_0}...\epsilon^i_{n_n}]$. Finalment, per complir la condició de ϵ mínima, es pot calcular aquest valor per a cada freqüència fent:

$$\epsilon_i = A_W^T \cdot [A_W \cdot A_W^T]^{-1} \cdot B \tag{6.5}$$

i es busca, entre totes les freqüències, la mínima o zero per trobar ϵ^- i la màxima o zero per trobar ϵ^+.
Un cop realitzats aquests càlculs es troben els coeficients del model intervalar, i es procedeix a validar-lo realitzant un diagrama de Bode i veient com les combinacions amb els valors màxims i mínims embolcallen perfectament les dades experimentals de partida. Tot i que aquesta validació es produeix a "simple vista", més endavant es detallarà un mètode més acurat per a aquest embolcallament.

6.2 Identificació robusta de *Pitch*

6.2.1 Obtenció del model nominal de partida G_0

En el capítol anterior s'ha procedit a identificar, a partir d'un assaig dades experimentals, el model dinàmic de l'angle de *Pitch* del *Dron*.
S'anomena model nominal al model ja identificat:

$$FT_{final} = \frac{-131685.11 - 8586.46s + 63.16s^2 + 30.05s^3}{-2.28 + 35.56s - 5.10s^2 + s^3} = G_{Pitch,Vx} \tag{6.6}$$

El nou model serà del mateix ordre que el primer, per tant, tindrà els coeficients $\hat{n_i}$ i $\hat{d_i}$ fins a ordre 3,3.

$$\mathbf{G}(s) = \frac{\hat{n_3}s^3 + \hat{n_2}s^2 + \hat{n_1}s + \hat{n_0}}{\hat{d_3}s^3 + \hat{d_2}s^2 + \hat{d_1}s + 1} \tag{6.7}$$

6.2.2 Identificació dels coeficients intervalars del sistema

Cada model que es determinarà consta de N dades, tantes com el nombre de freqüències de l'assaig. Per tant, primer es creen els N models cridant la funció

$$[\mathrm{H},\mathrm{B},\mathrm{A},\mathrm{w}] \;=\; \mathrm{identifica}\,(\mathrm{H},\mathrm{f}\,,\mathrm{na}\,,\mathrm{nb})$$

(funció creada per l'estudiant) que agafa les dades H i troba els vectors de coeficients A i B d'un polinomi d'ordres na i nb. Modificant una d'aquestes dades s'aconsegueix un model diferent cada vegada, és a dir, $\mathbf{G}^I(s)$. S'ordenen els coeficients en un vector tal que

$$p(i) = [n_0 \;\; n_1 \;\; n_2 \;\; n_3 \;\; d_0 \;\; d_1 \;\; d_2 \;\; d_3] \tag{6.8}$$

i se'n calcula les sensibilitats $\frac{\partial p}{\partial \mathbf{G}^I(j\omega)}$ per omplir la matriu. Del promig, se'n treuen els pesos i les $\epsilon^{\pm}$ per a cada coeficient del model, buscant-ne el màxim cada vegada:

$$w \cdot \epsilon = [15325.55 \;\; 3571.39 \;\; 296.24 \;\; 22.36 \;\; 1.20 \;\; 2.79 \;\; 0.28 \;\; 1] \tag{6.9}$$

Aquests resultats corresponen a com d'ample serà l'interval al voltant del valor del paràmetre original. Per tant, proporcionen un model intervalar tal que

$$\mathbf{G}_I(s) = \frac{[7.7, 52.4]s^3 + [-233.1, 359.4]s^2 + [-12158.0, -5015.1]s + [-147010.0, -116360]}{s^3 + [-5.4, -4.8]s^2 + [32.8, 38.4]s + [-3.5, -1.1]} = G_{Pitch,Vx} \tag{6.10}$$

Resposta freqüencial del model intervalar resultant

Per validar aquest model, la opció més senzilla correspon a agafar aquests mateixos valors màxims de l'interval, i observar-ne si, representats en el diagrama de Bode, són capaços d'embolcallar les dades experimentals a simple vista. Per exemple,

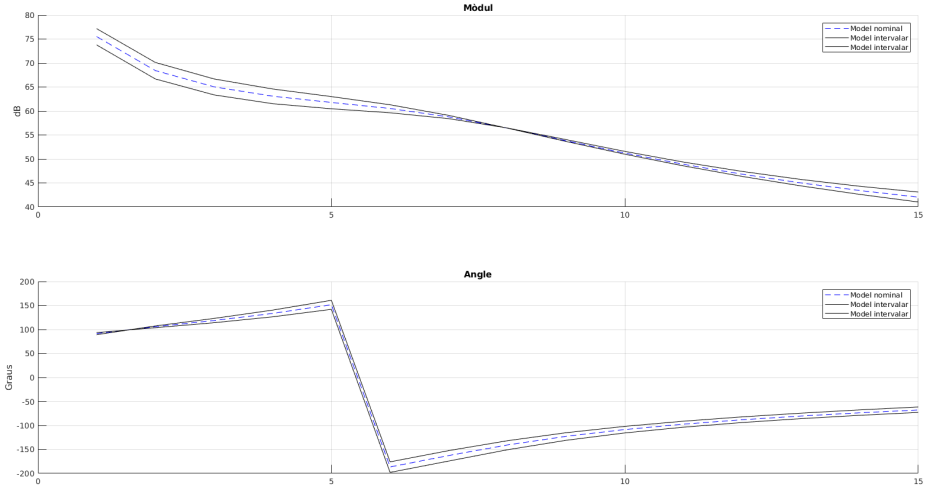


Figura 6.1: *Diagrama de Bode del model nominal de l'assaig p01 _ 01r4.mat i els models embolcallants*

Es pot veure, doncs, que el model queda perfectament embolcallat i que s'hauria de contemplar dins d'aquest marge tota la dinàmica del *Dron* per a l'angle de *Pitch*. Cal recordar que l'objectiu és identificar-ne un marge d'actuació per després trobar controladors que puguin actuar dins d'aquest marge i estabilitzar el *Dron* sota petites variacions de la dinàmica.

Resposta del model intervalar optimitzat

Si es desitja un mètode de validació més acurat es pot recórrer a la *Toolbox* de MATLAB d'optimització, i amb la funció

```
IntervalModelFrequencyResponse(n,N,d,D,w,Hini)
```

aconseguir embolcallar les dades més restrictivament. Aquesta funció utilitza l'eina `fmincon`, un algoritme que resol problemes no linials. El problema plantejat és el següent:

- Hi ha 8 paràmetres a determinar en el model de *Pitch*
- S'introdueixen 15 dades experimentals, a les corresponents freqüències
- S'especifiquen les restriccions que han de complir aquests paràmetres
- S'introdueix una llavor, un punt on el programa comença a iterar

Aleshores el programa passa les dades al diagrama de Nyquist, i crea una "caixa" al seu voltant, és a dir, un quadrat definit pels màxims i mínims valors reals i imaginaris que pot prendre el model a aquella freqüència concreta. La dada d'origen, per tant, no pot sortir-se d'aquesta caixa. La iteració segueix pel mètode del gradient, provant direccions i recomposant aquests costats del quadrat fins a optimitzar-los per a totes les dades experimentals i tots els paràmetres del model. És important que la llavor comenci des d'un punt adequat, ja que el programa pot estar hores corrent o no trobar solució. Per això s'ha donat com a llavor els valors del model nominal trobat en el capítol anterior.
Un exemple de com es veuria aquest diagrama de Nyquist amb 15 dades és el següent:

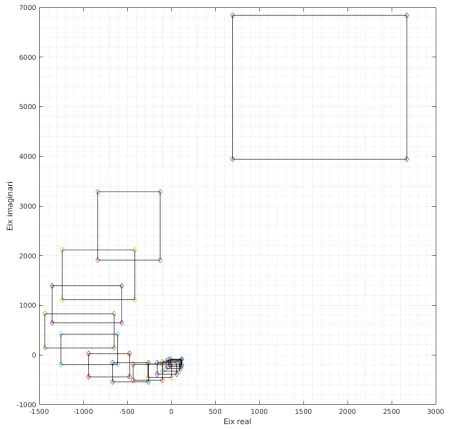


Figura 6.2: *Diagrama de Nyquist de l'assaig p01 _ 01r4.mat*

Alhora, si els resultats s'expressen en un diagrama de Bode és veu molt més intuitivament com els models intervalars que acaba determinant el programa embolcallen tant el model nominal (el que s'ha donat com a llavor) com les dades experimentals.

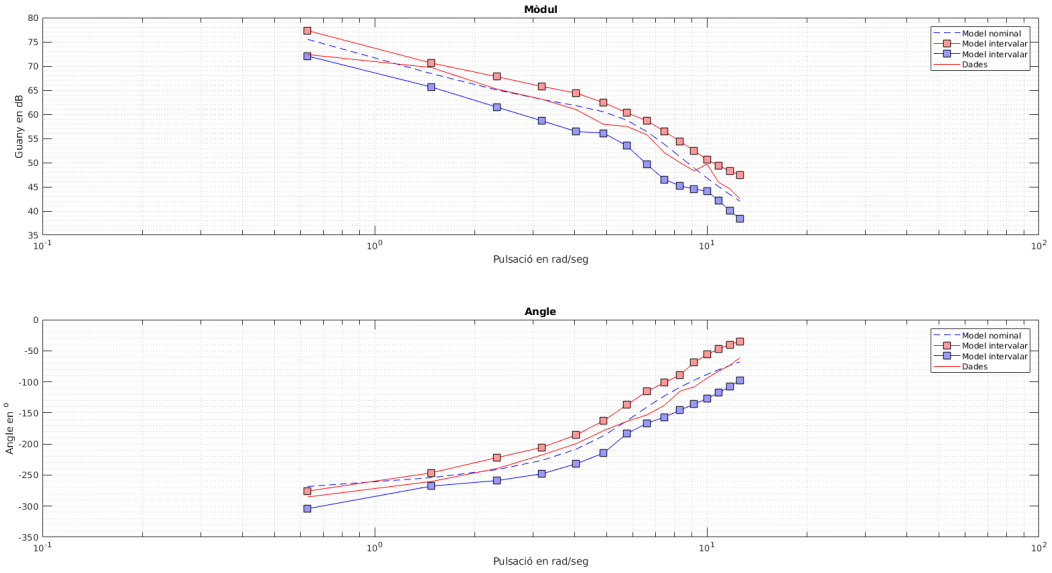


Figura 6.3: *Diagrama de Bode de l'assaig p01 _ 01r4.mat i el model intervalar embolcallant més restrictiu*

Finalment, el model intervalar identificat per l'angle de *Pitch* a partir de les dades experimentals de l'assaig *p01_ 01r4.mat* és:

$$\mathbf{G}_I(s) = \frac{[52.3, 93.9]s^3 + [-55.2, 550.7]s^2 + [-9193.5, -2678.7]s + [-140760, -107470]}{s^3 + [-6.8, -5.7]s^2 + [31.3, 38.9]s + [-11.1, -7.9]} = G_{Pitch,Vx} \tag{6.11}$$

6.3 Identificació robusta de *Roll*

6.3.1 Obtenció del model nominal de partida G_0

En el cas de l'angle de *Roll* el model prèviament identificat és:

$$FT_{final} = \frac{19715.07 + 1444.48s + 212.20s^2}{2.72 - 3.05s + s^2} = G_{Roll,Vy} \tag{6.12}$$

El nou model serà del mateix ordre que el primer, per tant, tindrà els coeficients $\hat{n}_i$ i $\hat{d}_i$ fins a ordre 2,2.

$$\mathbf{G}(s) = \frac{\hat{n_2}s^2 + \hat{n_1}s + \hat{n_0}}{\hat{d_2}s^2 + \hat{d_1}s + 1} \tag{6.13}$$

6.3.2 Identificació dels coeficients intervalars del sistema

Ídem que en el cas del *Pitch*, el promig de les sensibilitats se'n treuen els pesos i les $\epsilon^{\pm}$ per a cada coeficient del model, buscant-ne el màxim cada vegada:

$$w \cdot \epsilon = [808.87 \;\; 190.90 \;\; 15.47 \;\; 0.33 \;\; 0.20 \;\; 0] \tag{6.14}$$

Aquests resultats corresponen a com d'ample serà l'interval al voltant del valor del paràmetre original. Per tant, proporcionen un model intervalar tal que

$$\mathbf{G}_I(s) = \frac{[196.73, 227.68]s^2 + [1253.58, 1635.37]s + [18906.11, 20523.84]}{s^2 + [-3.25, -2.84]s + [2.39, 3.05]} = G_{Roll,Vy} \tag{6.15}$$

Resposta freqüencial del model intervalar resultant

Agafant els valors màxims de l'interval i representant-los en un diagrama de Bode:

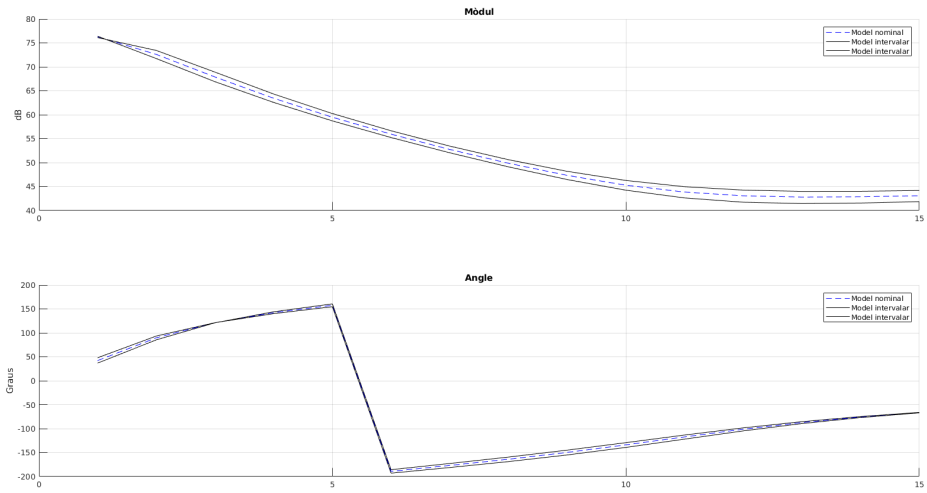


Figura 6.4: *Bode del model nominal de l'assaig r01 _ 01r2.mat i els models embolcallants*

Altra vegada, el model queda perfectament embolcallat, però les dades experimentals no. Caldrà recórrer a l'algoritme d'optimització dels coeficients.

Resposta del model intervalar optimitzat

En aquest cas el programa *fmincon* ha d'optimitzar 6 paràmetres, ja que el model nominal de *Roll* és d'ordre 2,2. Així, el nombre de dades es mantenen, i les restriccions també. Aplicant la llavor del model nominal s'ha obtingut el següent model intervalar:

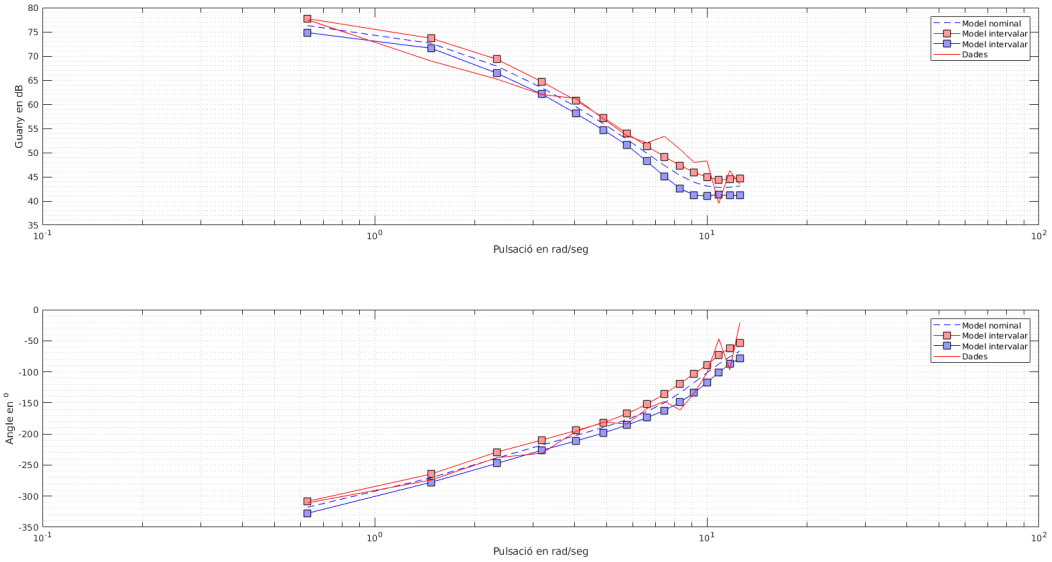


Figura 6.5: *Diagrama de Bode de l'assaig r01 _ 01r2.mat i el model intervalar embolcallant d'ordre 2,2 més restrictiu*

Es pot veure com aquests nous models no embolcallen bé, les dades experimentals queden fora. Pot ser que tot i la justificació que s'ha fet de l'ordre 2,2 del model de *Roll* mitjançant Akaike, aquest ordre no sigui capaç de trbar un model embolcallant. Per solucionar-ho, es repeteix la identificació del model nominal $G_0(s)$ amb ordres 3,3, es torna a calcular el model intervalar de Bhattacharyya i es torna a executar la rutina d'optimització. El resultat és el següent:

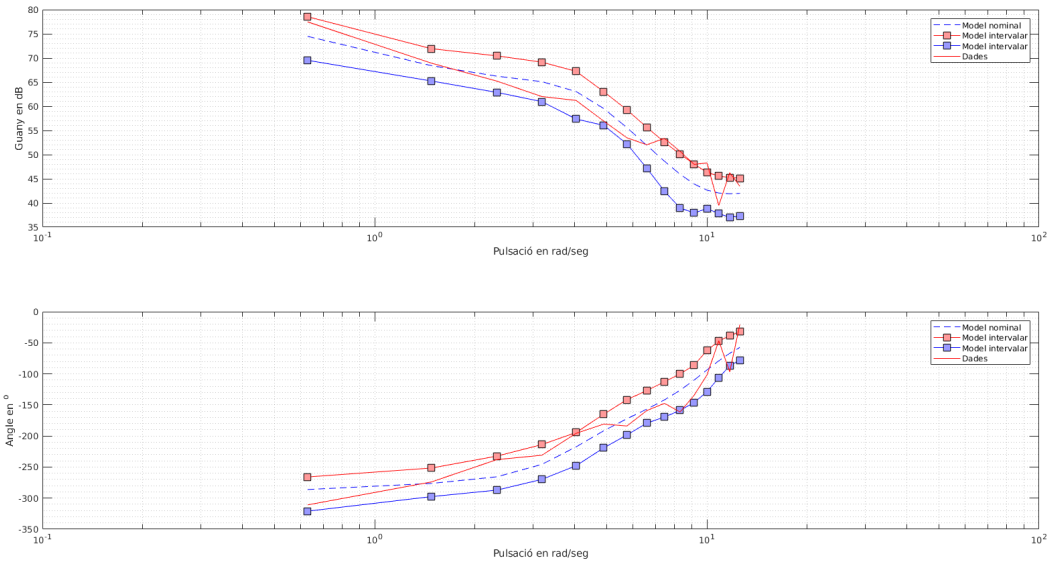


Figura 6.6: *Diagrama de Bode de l'assaig r01 _ 01r2.mat i el model intervalar embolcallant d'ordre 3,3 més restrictiu*

El resultat és acceptable en les freqüències baixes en les que es treballa en aquest estudi. Finalment, els models nominal i intervalar identificats per l'angle de *Roll* a partir de les dades experimentals de l'assaig *r01_ 01r2.mat* són:

$$\mathbf{G}_0(s) = \frac{176.08s^3 + 595.69s^2 + 13587.00s - 55760}{s^3 - 3.54s^2 + 17.28s - 2.80} = G_{Roll,Vy} \tag{6.16}$$

$$\mathbf{G}_I(s) = \frac{[160.74, 191.42]s^3 + [298.39, 892.99]s^2 + [12432.00, 14742.00]s + [-68112.00, -43407.00]}{s^3 + [-3.82, -3.27]s^2 + [14.78, 19.78]s + [-5.37, -0.22]} \tag{6.17}$$

6.4 Identificació robusta de *Yaw*

6.4.1 Obtenció del model nominal de partida G_0

En el cas de l'angle de *Yaw* el model prèviament identificat és:

$$FT_{final} = \frac{-128.08 + 18.48s + 2.30s^2}{0.97 - 0.55s + s^2} = G_{Yaw,\psi} \tag{6.18}$$

El nou model serà del mateix ordre que el primer, per tant, tindrà els coeficients $\hat{n}_i$ i $\hat{d}_i$ fins a ordre 2,2.

$$\mathbf{G}(s) = \frac{\hat{n_2}s^2 + \hat{n_1}s + \hat{n_0}}{\hat{d}_2s^2 + \hat{d}_1s + 1} \tag{6.19}$$

6.4.2 Identificació dels coeficients intervalars del sistema

Ídem que en el cas del *Pitch* i *Roll*, el programa `bhattacharyya.m` proporciona un model intervalar tal que

$$\mathbf{G}_I(s) = \frac{[2.02, 2.57]s^2 + [15.68, 21.28]s + [-139.47, -116.70]}{s^2 + [-0.74, -0.36]s + [0.87, 1.07]} = G_{Yaw,\psi} \tag{6.20}$$

Resposta freqüencial del model intervalar resultant

Agafant els valors màxims de l'interval i representant-los en un diagrama de Bode:

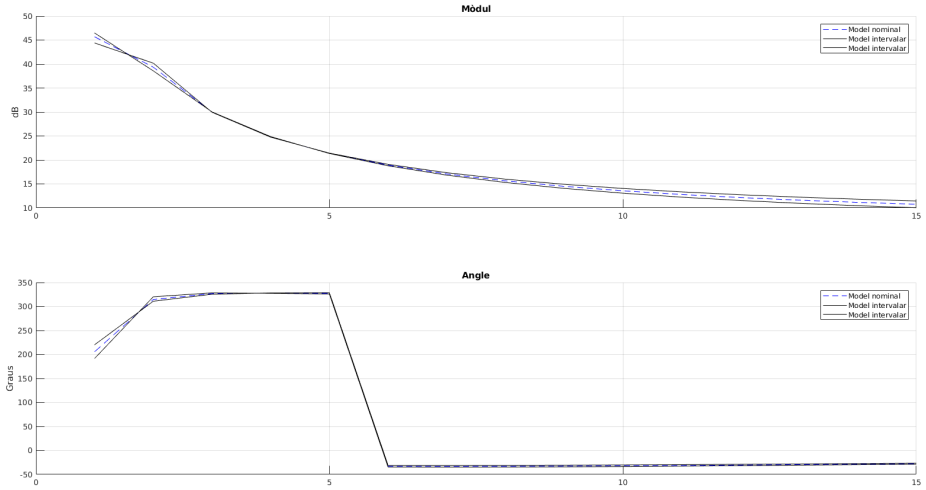


Figura 6.7: *Bode del model nominal de l'assaig y01 _ 01r2.mat i els models embolcallants*

Altra vegada, el model queda perfectament embolcallat, però les dades experimentals no. Caldrà recórrer a l'algoritme d'optimització dels coeficients.

Resposta del model intervalar optimitzat

En aquest cas el programa *fmincon* ha d'optimitzar 6 paràmetres, ja que el model nominal de *Yaw* és d'ordre 2,2. Així, el nombre de dades es mantenen, i les restriccions també. Aplicant la llavor del model nominal s'ha obtingut el següent model intervalar:

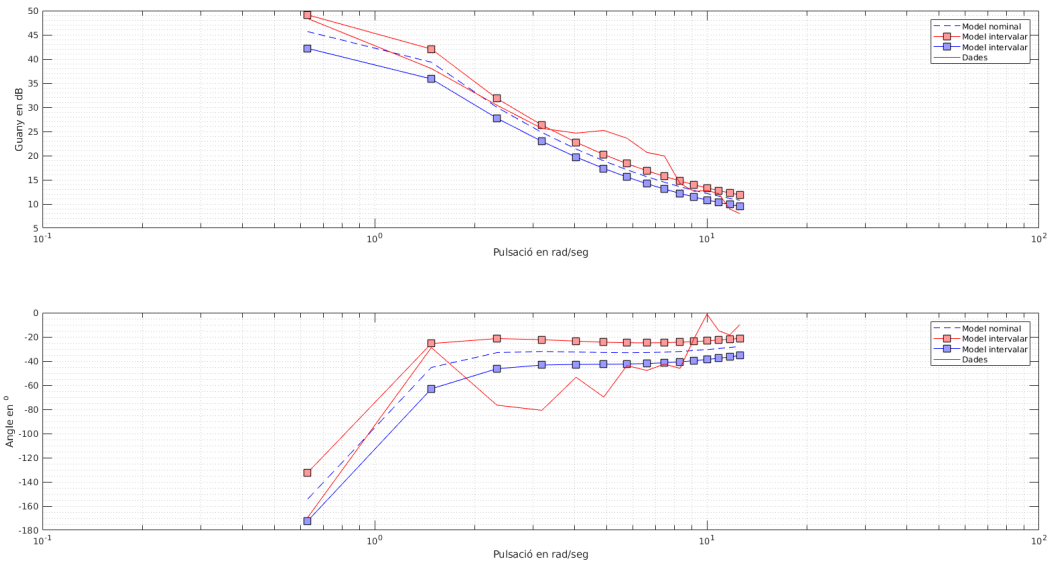


Figura 6.8: *Diagrama de Bode de l'assaig y01 _ 01r2.mat i el model intervalar embolcallant més restrictiu*

$$\mathbf{G}_I(s) = \frac{[1.20, 2.73]s^2 + [16.00, 21.47]s + [-139.40, -115.78]s}{[0.56, 1.44]s^2 + [-2.09, -0.07]s + [0.62, 1.69]} \tag{6.21}$$

Es pot veure com aquests nous models no embolcallen bé, les dades experimentals queden fora. Pot ser que tot i la justificació que s'ha fet de l'ordre 2,2 del model de *Yaw* mitjançant Akaike, aquest ordre no sigui capaç de trbar un model embolcallant. No obstant, s'ha provat amb ordres superiors i tampoc embolcalla bé. Es pot mirar de trobar controladors amb aquest model, que a freqüències baixes sí que funciona.

Capítol 7

Disseny dels controladors intervalars robusts

Un model intervalar contempla la incertesa que pot patir una planta deguda a simplificacions assumides o imprecisions a l'hora d'identificar el model. Per tant, en el moment de calcular el controlador adient s'ha de contemplar tota aquesta casuística i assegurar que funciona per a tota la família de models trobada. A més, segons la dinàmica que es vulgui aconseguir, s'haurà de sintetizar un controlador o un altre. És per aquesta raó que s'ha desenvolupat un mètode per calcular una família de controladors PI que funcionin amb els models intervalars ja trobats i que permetin assolir dinàmiques relativament diferents.
El mètode parteix d'un primer pas: determinar les 16 plantes de Barmish, que no són altra cosa que la combinació de quatre polinomis de Kharitonov del numerador amb quatre polinomis de Kharitonov del denominador. Aquestes plantes seràn de la forma $G(s) = \frac{B(s)}{A(s)}$ i pretenen calcular controladors de la forma $C(s) = \frac{K_p s + K_i}{s}$.

7.1 Les 16 plantes de Barmish

Partint d'un model intervalar es poden combinar els coeficients per obtenir 16 plantes diferents. Cada planta defineix una regió d'estabilitat, per tant, la intersecció de totes elles proporciona la regió d'estabilitat de la família de models totals. Així, el teorema diu que

Un controlador estabilitza de manera robusta una família de plantes si, i només si, estabilitza cadascuna de les 16 plantes de Barmish

Donat $B(s,b) = \sum_{i=0}^{nb} [b_i^-; b_i^+] s^i$ es calculen els polinomis de Kharitonov com

$$B_1(s) = b_0^- + b_1^- s + b_2^+ s^2 + b_3^+ s^3 + ... \tag{7.1}$$

$$B_2(s) = b_0^+ + b_1^+ s + b_2^- s^2 + b_3^- s^3 + ... \tag{7.2}$$

$$B_3(s) = b_0^+ + b_1^- s + b_2^- s^2 + b_3^+ s^3 + ... \tag{7.3}$$

$$B_4(s) = b_0^- + b_1^+ s + b_2^+ s^2 + b_3^- s^3 + ... \tag{7.4}$$

Ídem per A.
Realitzant variacions amb repetició dels polinomis calculats s'obtenen 16 plantes diferents.

7.2 Síntesi de controladors robusts de Barmish

El següent pas és construir una taula de Routh per cadascuna de les 16 plantes.

Criteri de Routh

Per determinar l'estabilitat d'un model del qual el denominador en llaç obert o tancat és un polinomi, es pot seguir el criteri de Routh [24], pel qual es construeix una taula amb els coeficients d'aquest polinomi, començant pel més alt i reduint l'ordre d'1 en 1 cap a la dreta i de 2 en 2 cap a baix. La condició necessària i suficient per a que el sistema sigui qualificat d'estable és que els coeficients de tota la primera columna tinguin el mateix signe. Cada vegada que hi hagi un canvi de signe, i per tant, sigui inestable, hi ha un pol més de part real positiva.

Per tant, la primera columna de la taula depèn dels paràmetres K_p i K_i, i el criteri d'estabilitat de que siguin positius proporciona un seguit d'inequacions que, graficades, delimitaran la regió d'estabilitat.
S'ha utilitzat una funció ja programada en un treball anterior [11] anomenada `Barmish.m` que tracta K_p i K_i com a variables simbòliques, genera les taules de Routh i en treu les rectes per graficar-les. Al córrer el programa es apareixen les 16 línies en color verd, i un missatge per seleccionar un punt de l'espai restringit amb l'eix d'abscises, que donarà una parella de valors dels controladors a utilitzar. Per exemple:

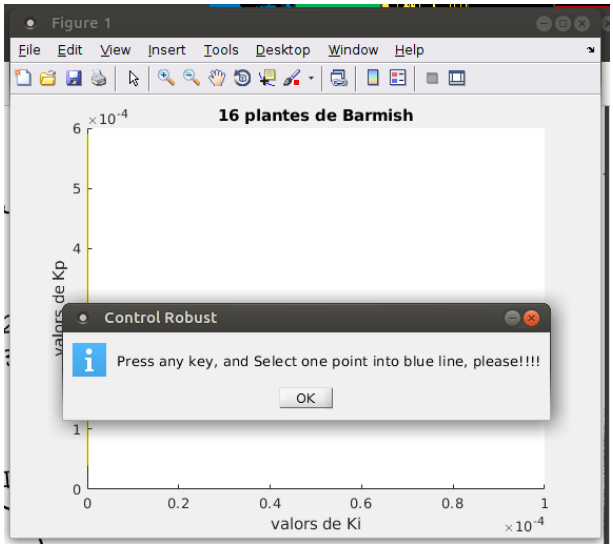


Figura 7.1: *Exemple d'execució de la rutina Barmish.m*

7.2.1 Controlador robust de *Pitch*

En el cas de l'angle de *Pitch* la regió indicada queda entre la recta $K_i = 0$ i la primera que es veu a la figura:
La resta de les altres 16 rectes limitants que no es veuen arribaven a valors de Ki de quasi 3000. S'ha apropat la imatge a les rectes que interessen, les que queden més a l'esquerra per trobar la

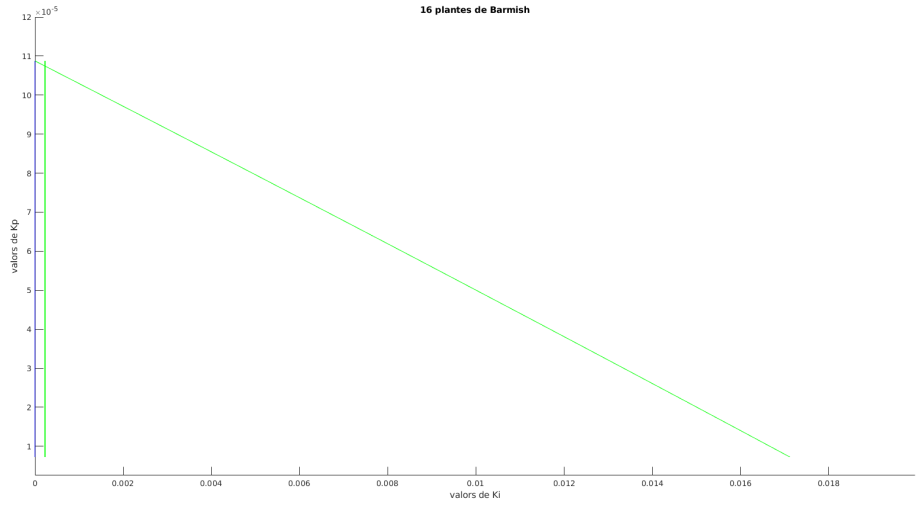


Figura 7.2: *Regió d'estabilitat dels controladors robusts de Pitch*

regió limitada amb K_i (recta de color blau). Es pot veure, doncs, que la zona de controladors desitjats es trobaria, agafant sempre valors positius, en

$$0 < K_p < 1.1 \cdot 10^{-4} \qquad 0 < K_i < 2 \cdot 10^{-4}\ seg^{-1}$$

Teòricament, agafant qualsevol parella de valors (K_p^*, K_i^*) s'hauria d'estabilitzar el model de *Pitch*, però obtenint dinàmiques diferents.
Com s'ha provat al capítol de Disseny de controladors sobre el model nominal, els valors de

$$K_p = 0.000057825 \quad K_i = 0.0000011015\ seg^{-1}$$

es troben dins la regió seleccionada pel mètode de Barmish, així que són controladors vàlids per al model de *Pitch* intervalar ja identificat. Si es desitja una dinàmica més agressiva es poden variar els paràmetres, sempre mantenint-se dins la regió especificada.

7.2.2 Controlador robust de *Roll*

Igual que abans, la regió indicada queda entre la recta $K_i = 0$ i la primera que es veu a la figura:

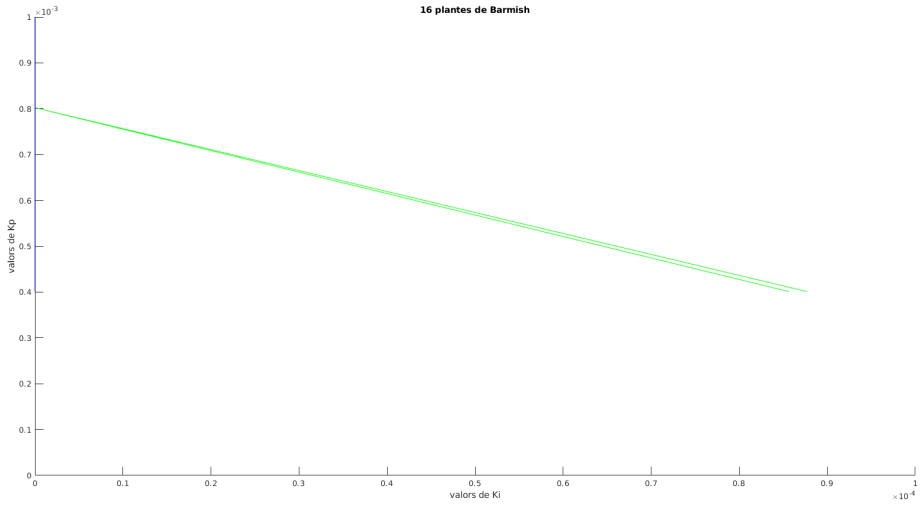


Figura 7.3: *Regió d'estabilitat dels controladors robusts de Roll*

Es pot veure, doncs, que la zona de controladors desitjats es trobaria, agafant sempre valors positius, en

$$0 < K_p < 6 \cdot 10^{-4} \qquad 0 < K_i < 9 \cdot 10^{-4}\ seg^{-1}$$

7.2.3 Controlador robust de *Yaw*

Igual que abans, la regió indicada queda entre la recta $K_i = 0$ i la primera que es veu a la figura: Es pot veure, doncs, que la zona de controladors desitjats es trobaria, agafant sempre

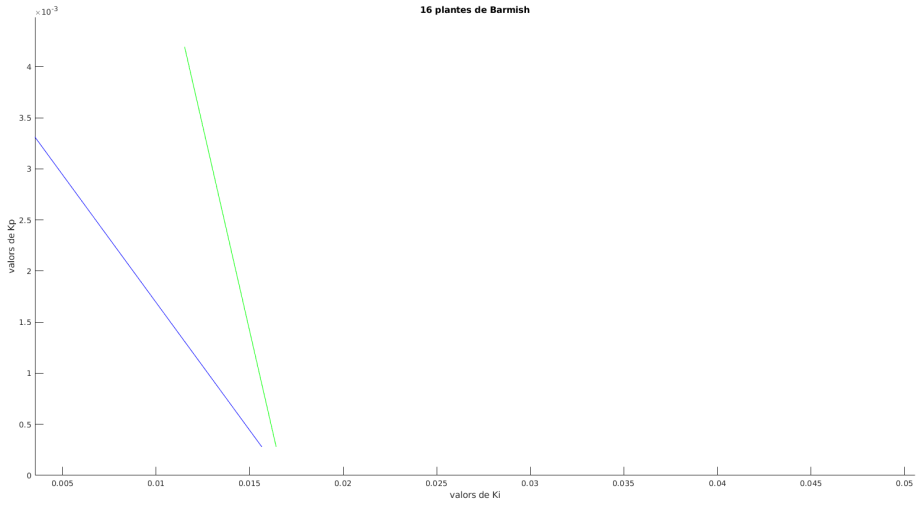


Figura 7.4: *Regió d'estabilitat dels controladors robusts de Yaw*

valors positius, en

$$0 < K_p < 3 \cdot 10^{-3} \qquad 0 < K_i < 0.015\ seg^{-1}$$

Part III

Visió

Capítol 8

Adquisició i calibració d'imatge de la càmara zenital

8.1 Adquisició

Dins el node de ROS que representa el *Dron*, es pot cridar una eina per establir el canal de comunicació amb la càmara, i una altra per canviar entre càmara frontal i zenital:

```
setCamFrontCLI = rossvcclient('/ardrone/setcamchannel')
toggleCamCLI = rossvcclient('/ardrone/togglecam')
```

Quan s'inicialitza el *Dron*, es subscriu al canal corresponent per rebre imatges, les quals es reben en format *raw*

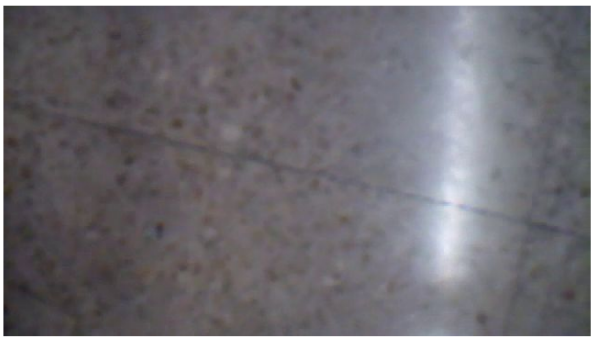
```
imageSUB = rossubscriber('/ardrone/image\_raw/compressed')
```

Un cop iniciat el moviment, si es volen rebre imatges s'ha de cridar al canal subscrit i llegir-les amb una eina de la *Robot System Toolbox* de MATLAB per processar-les posteriorment.

```
imreceived=receive(imageSUB)
image=readImage(imreceived)
imshow(image)
```

Utilitzant aquestes comandes dins de la rutina del temporitzador es pot anar seguint el que veu la càmara des de la pantalla de l'ordinador. Més endavant, es podrà comprobar com fa el seguiment d'una línia en el terra, i si en algun moment la perd de la vista de la càmara.

(a) *Càmara frontal*



(b) *Càmara zenital*

El primer pas ha estat, per tant, obtenir imatges *raw* tant de la càmara frontal com de la zenital, fetes a l'entorn on es farà treballar el *Dron*, i prosseguir a la calibració de la càmara zenital.

El *Parrot* disposa d'una càmara zenital que captura imatges en color de 1280x720 píxels a 30fps. L'objectiu és el de seguir una línia pintada al terra mitjançant les imatges recollides per aquesta càmara. Per tant, en experiments previs es va triar el color vermell de línia ja que es treballa en un terra de rajoles amb tons grisos, blancs i negres.
El problema de l'adquisició rau en relacionar una posició de la imatge capturada amb una posició del terra a la realitat. Cal fer doncs, transformacions d'eixos i calibració de la càmara, que es recolliran en una funció de MATLAB que anomenarem `calibracio.m`.

8.2 Transformacions d'eixos

El primer canvi es realitzarà entre el sistema de coordenades del *Dron* i el de la càmara. El primer s'orienta en eixos terra, i només cal rotar X i Y 90 graus amb la matriu:

$$\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

En aquest cas no cal translació d'eixos, per tant la matriu de translació serà la identitat.

8.3 Calibració

S'ha seguit la guia de calibració de

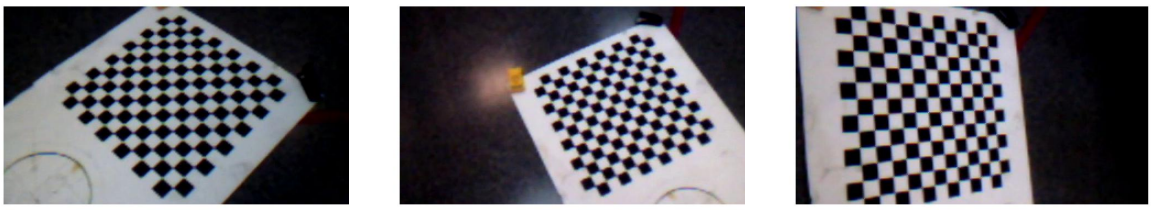
$$http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html$$

Per transformar un punt en coordenades de la imatge a un punt en coordenades del *Dron* i guiar-lo amb elles, s'ha de determinar els paràmetres intrínsecs i extrínsecs de la càmara.

8.3.1 Paràmetres intrínsecs

- Distància focal `fc`
- Punt principal `cc`
- Coeficient d'inclinació `alphac`
- Distorsions `kc`

Dins la *Toolbox* es demana processar una imatge treta amb la càmara del *Dron* d'un escaquer, on es guia al programa per detectar les cantonades dels quadrats de la imatge.



(a) *Imatge 1* (b) *Imatge 2* (c) *Imatge 3*

Figura 8.2: *Detecció dels paràmetres instrínsecs*

Aquesta eina proporciona tots els paràmetres intrínsecs esmentats, que s'emmagatzemen en la matriu KK com:

$$\begin{bmatrix} fc(1) & \alpha_c * fc(1) & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 570.3 & 0 & 412.5 \\ 0 & 612.4 & 225.0 \\ 0 & 0 & 1 \end{bmatrix}$$

8.3.2 Paràmetres extrínsecs

Una vegada són coneguts els paràmetres intrínsecs de la càmara, es realitza una captura d'imatge de l'escaquer frontalment. S'agafa un punt de referència de l'escaquer i se'n mesura la distància fins al punt mig del *Dron*, i també se'n mesura l'alçada. S'introdueixen les coordenades homogènies d'aquest punt a

$$Prob = [-62.7 \quad 104.9 \quad 86.0 \quad 1]$$

(en centímetres). S'emmarquen dins de la imatge un nombre concret de quadrats, se li dóna al programa la dimensió d'aquests quadrats (50 mm) i, coneguts els paràmetres intrínsecs, el programa en detecta les cantonades amb una certa incertesa.

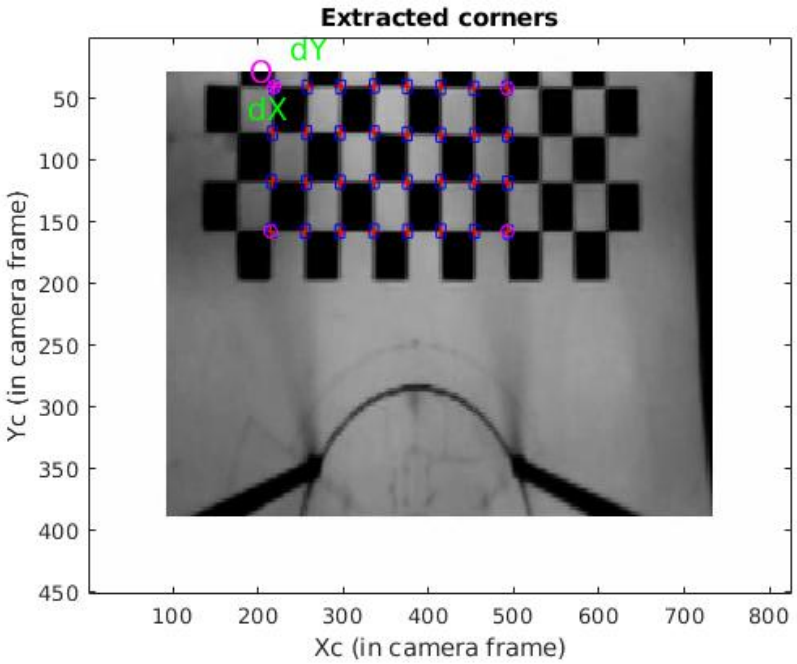


Figura 8.3: *Detecció dels quadrats*

Finalment, amb aquestes dades és capaç d'identificar els eixos de la imatge. Es pot veure que són diferents dels eixos del *Dron*, per això serà necessària la matriu de rotació abans esmentada.

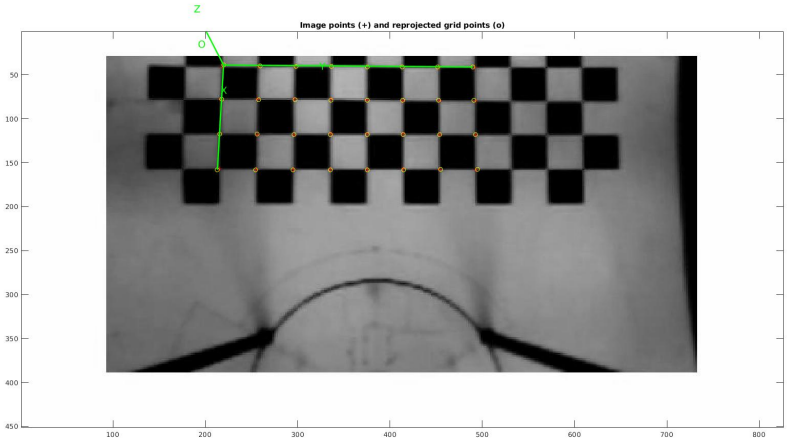


Figura 8.4: *Orientació dels eixos de la imatge*

Finalment, els resultats extrínsecs que permetran obtenir les coordenades són:

$$Rc_{ext} = \begin{bmatrix} 0.018483 & 0.998663 & 0.048274 \\ 0.971507 & -0.006528 & -0.236920 \\ -0.236288 & 0.051277 & -0.970329 \end{bmatrix}$$

$$Tc_{ext} = [-250.539201 \quad -223.702716 \quad 739.666858]$$

Capítol 9

Processat d'imatge de la càmara zenital

Per processar una imatge llegible des de MATLAB, i fer-la útil per a l'estudi cal aplicar una base colorimètrica adient segons el color que es vulgui visualitzar, i preparar al programa per reconèixer una línia dins la imatge tractada.
Agafant la imatge anterior de la càmara frontal es canvia primer a la base colorimètrica YC_bC_r (on Y correspon a la lluminància, C_b a la crominància blava i C_r a la crominància vermella) i, en aquest cas, s'eliminen totes les capes excepte la tercera. Així, en la imatge haurien d'aparèixer només les dades corresponents a la crominància vermella.

(a) YC_bC_r

(b) *Lluminància* Y

(c) *Crominància blava,* C_b

(d) *Crominància vermella,* C_r

Com que la línia a seguir serà vermella, interessa l'última transformació. Com s'aprecia en la figura de crominància vermella, C_r, els elements que no tenen component vermella es veuen en tons gris fosc, i els que sí que en tenen en tons gris clar. D'aquesta manera, s'ha creat una imatge en escala de grisos que es voldria transformar a "binària", o blanc o negre.
L'eina

```
imageBW = im2bw(imageCR, level)
```

permet "binaritzarl·la imatge en:

- els píxels que tenen lluminància més alta que el paràmetre `level` surten blancs (valor 1)
- la resta surten negres (valor 0)

(a) *Crominància vermella, C_r*

(b) *Blanc i negre*

En el cas de la figura anterior, s'ha aplicat un valor `level` força alt, de 0.65, ja que l'entorn tenia molta component vermella.

El següent pas és indicar-li a MATLAB què constitueix una línia, i comparar la imatge binària amb aquesta línia. Es crea un element estructural fent:

```
line = strel('line',long,degrees)
```

i s'erosiona la imatge perquè elimini elements no coincidents amb aquesta línia:

```
imageline = imerode(imageBW,line)
```

Per últim, el programa ha d'interpretar que està veient una línia. L'eina `bwmorph` aplica una operació demanada sobre una imatge binària. L'eina pot afegir o treure píxels per aprimar una imatge, o per definir-la millor. En aquest cas, la comanda `'skel'` elimina píxels dels límits d'un objecte sense deixar que es "trenquin".

```
imagemorph = bwmorph(imageline,'skel',Inf)
```

A cada instant de temps T_m, el programa ha de llegir la imatge que envia el *Dron* i identificar-ne línies verticals i horitzontals. Aquesta informació s'envia a la navegació perquè substitueixi a la lectura dels sensors, i faci de realimentació del llaç de posició.
Es parteix d'una imatge de la càmara zenital d'aproximadament 700mm d'amplada. Cada píxel de la imatge correspon a 0.93mm. Es considerarà una línia vertical com aquell element que

(a) *Blanc i negre*

(b) *Imatge erosionada*

creua per dalt i per baix de la imatge, i la horitzontal com la que creua per l'esquerra i la dreta de la imatge. Així, una línia vertical no sortirà del rang $-45 < \theta < 45$, i la horitzontal entre $-90 < \theta < 90$.
L'eina de MATLAB utilitzada és la *Transformada de Hough*, un algorisme destinat a reconèixer línies en una imatge binària, utilitzant coordenades polars: ρ i θ. Amb les comandes per a vertical:

```
[hV, thetaV, rhoV] = hough(imagemorphV,'Theta',-45:45)
peakV=houghpeaks(hV,1)
linesV = houghlines(imagemorphV,thetaV,rhoV,peakV)
```

i per a horitzontal:

```
[h, theta, rho] = hough(imagemorphH,'Theta',-90:89)
peak=houghpeaks(h,1)
linesH = houghlines(imagemorphH,theta,rho,peak)
```

Aquesta eina és de gran utilitat ja que proporciona la orientació i la distància a la que es troba la línia de l'extrem de la imatge. Es considera que el punt "mig"del rang de visió són 330mm. Així, la transformada proporciona els punts de la recta identificada i permet desenvolupar diferents accions segons la distància i orientació a la que es trobi. Per exemple:

(a) *Imatge original*

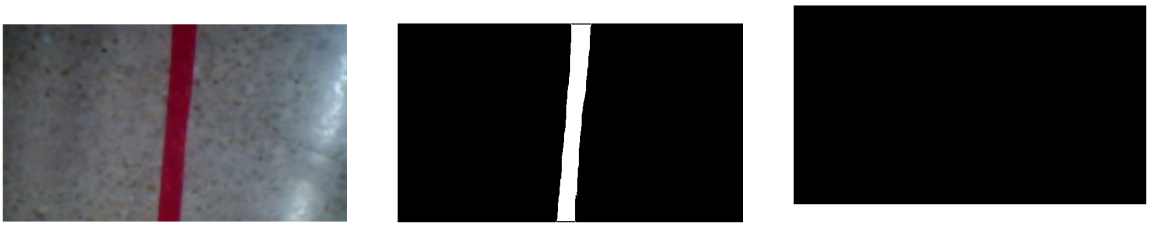
(b) *Imatge erosionada vertical*

(c) *Imatge erosionada horitzontal*

Figura 9.4: *Imatge del terra "blanc"*

	Línia vertical		**Línia horitzontal**	
	X	Y	X	Y
punt 1	0	0	0	0
punt 2	0	0	0	0

Taula 9.1: *Coordenades de les línies de la imatge del terra "blanc"*



(a) *Imatge original* (b) *Imatge erosionada vertical* (c) *Imatge erosionada horitzontal*

Figura 9.5: *Imatge de la línia vermella*

	Línia vertical		**Línia horitzontal**	
	X	Y	X	Y
punt 1	340	3	0	0
punt 2	297	358	0	0
ρ	337			.
θ	7			

Taula 9.2: *Coordenades de les línies de la imatge d'una línia vermella*

(a) *Imatge original* (b) *Imatge erosionada vertical* (c) *Imatge erosionada horitzontal*

Figura 9.6: *Imatge d'una cruïlla vermella*

	Línia vertical		**Línia horitzontal**	
	X	Y	X	Y
punt 1	301	88	18	100
punt 2	334	358	240	85
ρ	287		100	.
θ	-7		86	

Taula 9.3: *Coordenades de les línies de la imatge d'una cruïlla vermella*

Com es pot veure en aquest últim cas, a la línia horitzontal se li han restat molts més píxels que a la vertical, ja que durant la lectura de línies verticals la màquina podria confondre's i interpretar alguna part com a horitzontal. Això afectaria a la màquina d'estats que governa el *Dron*, que envia la comanda de seguir endavant si detecta línies verticals, i canvia de comanda si en detecta d'horitzontals.

Part IV

Planificació de trajectòries

Capítol 10

Disseny del controlador de trajectòria

Un cop obtinguts els paràmetres de possibles controladors per als tres angles de moviment (*Pitch*, *Roll* i *Yaw*, es poden utilitzar per l'aplicació final d'aquest treball.
Com es veu en la figura següent, es pretén tancar els llaços que s'han plantejat en la identificació obtenint imatges de la càmara zenital i utilitzant-les com a control de trajectòria per al llaç de posició. Així, a cada període (0.1 segons) el *Dron* rebrà la orientació a la que es troba respecte les línies del terra. La tasca següent és la de dissenyar un algoritme que interpreti

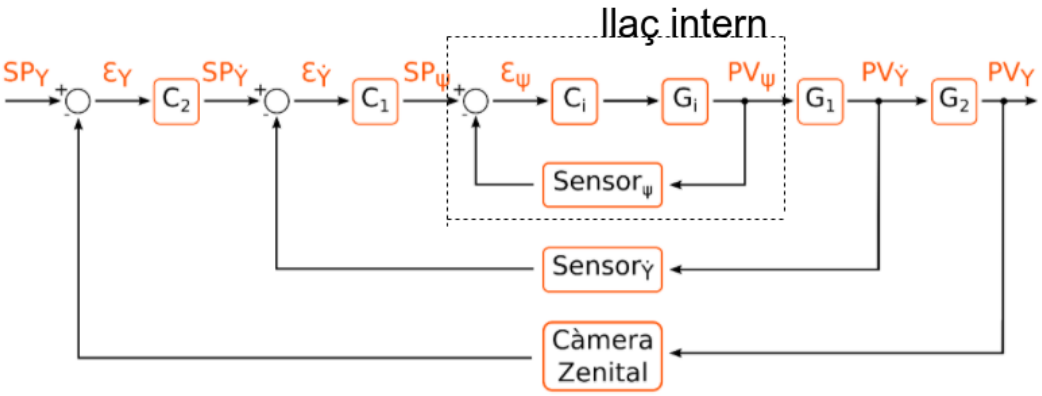


Figura 10.1: *Imatge extreta de [9]*

aquesta distància i intenti minimitzar-la, centrar-se sobre una línia vertical, avançar per ella, i finalment girar en detectar una intersecció de línia.

10.1 Màquina d'estats

Es crea una màquina d'estats que hauria d'obeïr el *Dron*. Aquestes comandes s'integren dins de la funció de controlador, i regeixen les seves actuacions segons si no veu cap línia, veu línies verticals, o verticals i horitzontals.

Primer estat: No es detecta cap línia

S'implementa un control del *Dron* per botonera, per a guiar-lo fins a sobre les línies que es volen detectar. S'ha dissenyat un controlador específic amb l'eina `getkey.m` de MATLAB, que detecta quines tecles es pitgen al teclat de l'ordinador.

Segon estat: Es detecta línia vertical

L'objectiu és detectar una línia vertical, orientar-se bé, i seguir-la. Per això, es llegeix a quina distància es troba la línia respecte l'eix de coordenades de la imatge, per donar angle de *Roll*, en quina inclinació, per donar angle de *Yaw* i, a més de la orientació, se li dóna angle de *Pitch* positiu per seguir-la.

Tercer estat: Es detecta línia horitzontal

Aquest estat necessita de l'anterior per funcionar. Només s'entén com a cruïlla si es segueix una línia vertical i, sense perdre-la, n'apareix una d'horitzontal.
La primera consigna és aturar el *Dron*.

Quart estat: L'aeronau gira

QUan s'ha detectat l'estat anterior, l'aeronau entra en un estat de girar 90 graus cap a la dreta. L'estat s'atura quan es considera que el gir està complet, i es retorna cap al Segon estat: seguiment de línia altra vegada.

Part V

Conclusions i Bibliografia

Capítol 11

Conclusions

Finalment, un cop desenvolupada tota la teoria sobre identificació i control robust, i la calibració i tractament d'imatge de la càmara, s'ha procedit a implementar-ho al *Dron*. L'objectiu final ha estat posar en pràctica tots els resultats obtinguts fent que l'aeronau reconegués un circuit vermell pintat al terra, fent seguiment de línia i girs a les interseccions.

11.1 Punts de partida

Es va partir d'una eina que ja funcionava, provinent del treball "*Control de Trajectòria en un Dron UAV*"[9], i que realitzava:

1. Comunicació amb la controladora del *Dron* mitjançant MATLAB i ROS, utilitzant la rutina `ARD_ ROS.m`
2. Model experimental de la dinàmica del *Dron* amb el mètode d'*Output Error* pels angles de *Pitch* i *Roll*
3. Tractament d'imatge per al reconeixement de línies vermelles al terra

11.2 Evolució de les tasques del treball

Com ja s'havia especificat en l'abast, les tasques desenvolupades han estat:

1. **Posada a punt de la plataforma**
2. **Identificació (un model)**
3. **Disseny del controlador de velocitat lineal**
4. **Adquisició i processat de les imatges de la càmara zenital**
5. **Disseny del controlador de trajectòria**
6. **Identificació robusta**
7. **Re-disseny dels controladors**

Un cop acabat el projecte, ja es pot dir que s'ha assolit tot l'abast que s'havia proposat. Veient els punts de partida descrits, s'ha depurat la rutina `ARD_ ROS.m` de comunicació amb la controladora, eliminant variables globals, reduint el nombre de funcions, i analitzant els temps d'execució de cada procediment.
També s'han trobat models experimentals nominals i sintetitzat controladors PID, tot i que durant el procés s'ha hagut d'aprendre diverses metodologies noves per rebaixar ordres dels controladors.
La feina més àrdua ha estat la de programar els algoritmes necessaris per la identificació robusta, tant el de Bhattacharyya com el d'optimització de models intervalars. La tasca d'escollir l'ordre pel model ha fet anar endavant i enrere en el procés d'identificació.
A més, s'ha realitzat una calibració prèvia de la càmara zenital per obtenir-ne paràmetres més fiables a partir de les imatges. Això fa que el guiatge del *Dron* sigui més segur, que es pugui realitzar el seguiment de línia a més velocitat, i que es puguin implementar moviments més complexos que en el treball anterior.

11.3 Propostes de millora

Tot i que en gran mesura, s'han complert els objectius, han quedar un parell de tasques pendents de millora. A més, noves idees han anat sorgint durant l'elaboració d'aquest estudi que podrien obrir noves línies de treball amb aquesta aeronau:

1. **Disseny del controlador de trajectòria**: S'ha dissenyat una màquina d'estats que segueix el *Dron* quan detecta línies. La intenció és millorar aquest algoritme per assolir un resultat més satisfactori.

2. **Planificació de trajectòries**: Com s'estudia a una assignatura optativa del Grau, aquest programa hauria d'optimitzar un recorregut a través d'aquestes interseccions de línies o nodes.

3. **Control de *Gaz***: Es podria implementar un control d'alçada, fer l'estudi com s'ha fet amb els tres angles, per aconseguir canviar el camp de visió de la càmara zenital.

4. **Model intervalar més conservador**: Es podria utilitzar una mostra més gran d'assajos, i realitzar la identificació robusta amb tots ells per aconseguir un model més ampli però conservador.

5. **Ús de la càmara frontal**: Es podria utilitzar la càmara frontal per complementar el guiatge de l'aeronau a través de l'espai. Resultaria en un moviment més precís i ràpid.

6. **Assajos amb bateries**: Durant la identificació es podria tenir en compte l'estat de la bateria, i tractar-ho com a una variable més, per aconseguir un model més acurat.

Bibliografia

[1] H. Akaike, *A new look at the statistical model identification*, Institute of Statistical Mathematics, Minato-ku, Tokyo, Japan, 1974.

[2] A. Baez, *Apunts de l'assignatura d' Helicòpters*, ESEIAAT, Universitat Politècnica de Catalunya, Terrassa, España, 2017.

[3] B. R. Barmish, *New Tools for Robustness of Linear Systems*, Department of Electrical and Computer Engineering, University of Wisconsin, United States, 1994.

[4] S.P. Bhattacharyya i J.S. Lew, *System Identification Using Interval Dynamic Models*, American Central Conference Baltimore, Maryland, United States, 1994.

[5] P-J. Bristeau, P. Martin, E. Salatin i N. Petit, *The Role of the Propeller Aerodynamics in the Model of a Quadrotor UAV*, Budapest, Hungary, 2009.

[6] D. del Campo, *Apunts de l'assignatura de Mecànica de Vol*, ESEIAAT, Universitat Politècnica de Catalunya, Terrassa, España, 2016.

[7] W. Dong, Guo-Ying Gu, X. Zhu i H. Ding, *Modeling and Control of a Quadrotor UAV with Aerodynamic Concepts*, Engineering and Technology International Journal of Aerospace and Mechanical Engineering, Vol. 7, No. 5, 2013.

[8] J. R. de França Arruda, *Multisine Multiexcitation in Frequency Response Function Estimation*, Universidade Estadual de Campinas, Campinas, Sao Paulo, Brazil, 1993.

[9] J. Gómez i Vega, *Control de Trajectòria en un Drone UAV*, ESEIAAT, Universitat Politècnica de Catalunya, España, 2016.

[10] F. J. Harris, *On the use of Windows for Harmonic Analysis with the Discrete Fourier Transform*, Naval Ocean Systems Center, San Diego, California, United States, 1978.

[11] O. Ibar, *Disseny i Aplicació de Reguladors PI Robustos fent ús de Models Intervalars d'ordre reduït*, EUETIT, Universitat Politècnica de Catalunya, Terrassa, España, 2003.

[12] N.L.M. Jeurgens, *Implementing a SIMULINK controller in an AR.Drone 2.0*, 2016.

[13] V. Krishnamurthy i V. Seshadri, *Model Reduction using the Routh Stability Criterion*, Indian Institute of Technology, Madras, India, 1978.

[14] E. C. Levy, *Complex-Curve Fitting*, Space Technology Labs., Los Angeles, California, United States, 1958.

[15] L. Ljung i K. Glover, *Frequency domain versus time domain methods in system identification*, 5th IFAC Symposium on Identification and System Parameter Estimation, Darmstadt, Federal Republic of Germany, 1979.

[16] L. Ljung, *Perspectives on System Identification*, Division of Automatic Control, Linköpings universitet, Linköping, Sweden, 2007.

[17] G. Martin, *Modelling and Control of the Parrot AR.Drone*, School of Engineering and Information Technology, Canberra, Australia, 2012.

[18] A. Masip-Àlvarez, *Coupled Multivariable Systems*, ESEIAAT, Universitat Politècnica de Catalunya, Terrassa, España, 2015.

[19] A. Masip-Àlvarez, *Transformacions i rotacions d'eixos. Aplicació pràctica sobre el sistema de visió del Robotino*, ESEIAAT, Universitat Politècnica de Catalunya, Terrassa, España, 2015.

[20] R. Matusu, R. Prokop i Z. Prokopova, *Simple Tuning of PI Controllers for Interval Plants*, Faculty of Manufacturing Technology, TUKE, Slovak Republic, 2012.

[21] Mathworks S.A., *Get Started with ROS*,

URL: https://es.mathworks.com/help/robotics/examples/get-started-with-ros.html

[22] J. Pestana, J.L. Sanchez-Lopez, I. Mellado-Bataller, C. Fu i P. Campoy, *AR Drone Identification and Navigation Control at CVG-UPM*, Centre for Automation and Robotics, Universidad Politécnica de Madrid, España, 2012.

[23] S. Piskorki, N. Brulez, P. Eline i F. D'Haeyer, *AR.Drone Developer Guide, SDK 2.0*, Parrot S.A., United States, 2012.

[24] J. Quevedo, *Apunts de l'assignatura de Control Automàtic*, ESEIAAT, Universitat Politècnica de Catalunya, Terrassa, España, 2015.

[25] Open Source Robotics Foundation, *Robot Operating System*,

URL: http://www.ros.org/